HEVs
haute école valaisanne
hochschule wallis
sciences de l'ingénieur

Filière Systèmes industriels

Orientation Infotronics

Diplôme 2006

Simon Rinaldi

Intelligent Objects (Contextual Bluetooth)

Professeur	Medard Rieder
Expert	Jean-Pierre Buttet

route du rawyl 47 – case postale 2134 – ch-1950 sion 2
tél. +41 (0)27 606 85 11 – fax +41 (0)27 606 85 15
e-mail: info.sion@hevs.ch – internet: www.hevs.ch

ISO 9001

Sion, le 24 novembre 2006

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Rinaldi, Simon 2006

Intelligent Objects
(Contextual Bluetooth)

Objectif

Réaliser un environnement de démonstration qui contient deux objets intelligents. Est appelé objet intelligent un appareil dans lequel un processeur Bluetooth est embarqué. Lorsqu'un téléphone portable est à une distance définie de l'objet intelligent, une connexion s'établit. Les propriétés et comportements mis à disposition par l'objet sont envoyés au téléphone portable. Il est possible dès cet instant de télécommander l'objet intelligent.

Résultats

Une maquette, le rez-de-chaussée d'une maison, contient les objets intelligents « Garage » et « Télévision ». Il est possible, grâce à l'application exécutée par le téléphone portable, d'établir une connexion avec chacun d'entre eux en fonction de la position du téléphone portable dans la maison. Lorsque la connexion est établie, l'objet intelligent envoie une liste de propriétés et de comportements au téléphone portable. La télécommande de l'objet intelligent fonctionne.

Mots-clés

Bluetooth, objets intelligents, Java, J2ME, portée

Ziel

Realisierung eines Demonstrators mit zwei intelligenten Objekten, wobei die intelligenten Objekte ein eingebettetes System mit einem Bluetooth Chip darstellt. Sobald ein Mobiltelefon in die Reichweite dieses Systems kommt, wird automatisch eine Verbindung aufgebaut. Die Funktionalität des intelligenten Objekts wird an das Mobiltelefon geschickt. Dadurch wird dem Mobiltelefon die Möglichkeit gegeben, das System fernzusteuern.

Resultate

Ein Modell eines Erdgeschosses eines Hauses beinhaltet zwei intelligente Objekte „Garage" und „Fernsehapparat". Anhand der im Mobiltelefon laufenden Software ist es möglich, den Fernsehapparat oder die Garage zu bedienen, entsprechend welches Gerät gerade in Reichweite ist. Sobald das intelligente Objekt in Reichweite ist, sendet es eine List der Funktionalität an das Mobiltelefon und kann dadurch bedient werden.

Schlüsselwörter

Bluetooth, intelligente Objekte, Java, J2ME, Reichweite

Table des matières

Table des figures

Table des codes

1. Introduction

Objectifs :

L'objectif de ce projet est de réaliser un environnement de démonstration qui contient deux objets intelligents. Est appelé objet intelligent un appareil dans lequel un processeur est embarqué pour permettre d'établir une connexion Bluetooth avec une interface utilisateur. L'interface utilisateur est un téléphone portable. Lorsque la connexion est établie, il est possible d'utiliser le téléphone portable pour télécommander l'objet intelligent.

La connexion doit s'établir uniquement lorsque l'utilisateur se trouve à une portée définie de l'objet intelligent.

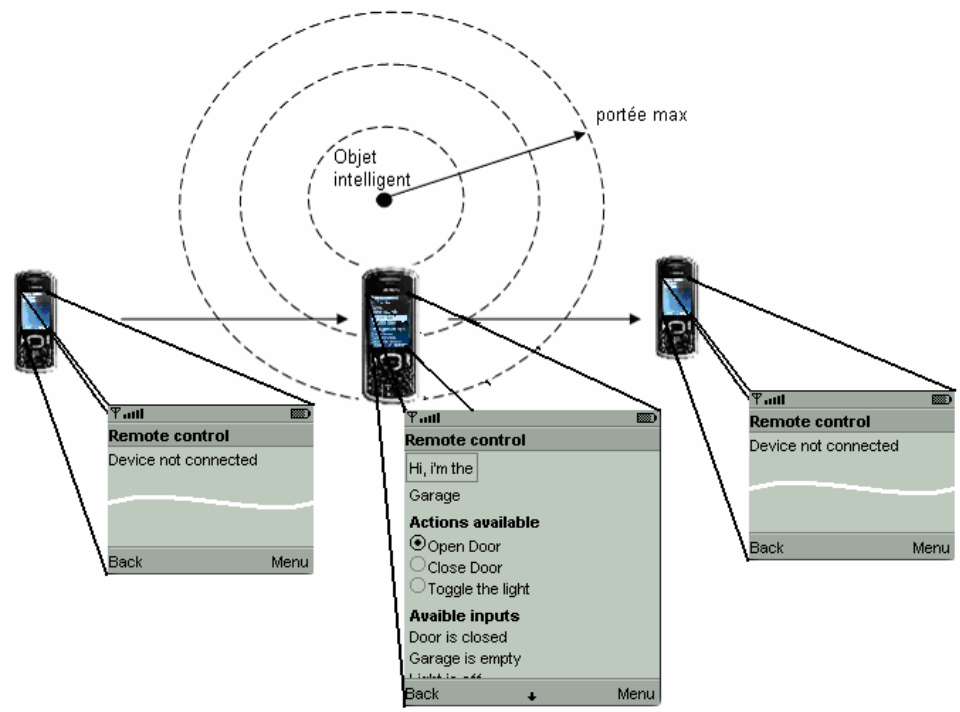


Figure 1 : Portée de l'objet intelligent

Étapes :

La première étape est d'établir dans quelle mesure il est possible de limiter la portée de l'objet intelligent. Ceci a pour but de le rendre inaccessible lorsque l'utilisateur s'éloigne au-delà d'une limite définie.

Lorsque ceci est établi, il faut réaliser l'objet intelligent. Cet objet a un comportement et des propriétés. Ceux-ci sont envoyés, sous forme de menu, à une interface utilisateur lorsque celui-ci s'approche de l'objet.

L'utilisation d'une maquette permet de montrer de manière très claire à un public les études réalisées lors du projet. Une maison miniature permet de montrer l'adaptabilité du projet à nos foyers.

2. Mesure de portée automatisée

Pour réaliser des mesures de portée, la première contrainte est de disposer de deux appareils équipés de la technologie Bluetooth.

Matériel à disposition :
- un kit de développement Casira (présenté en annexe B.1)
- un téléphone portable (présenté en annexe B.2)

Le kit de développement Casira permet la programmation d'un module amovible Bluetooth ainsi que d'accéder à des entrées/sorties numériques. Le téléphone portable me permet de créer une interface de mesure automatisée.

J'ai besoin de pouvoir varier quelques paramètres lors de ma mesure comme :

- le type de téléphone
- le type d'antenne de l'objet intelligent
- la puissance d'émission de l'objet intelligent
- l'environnement

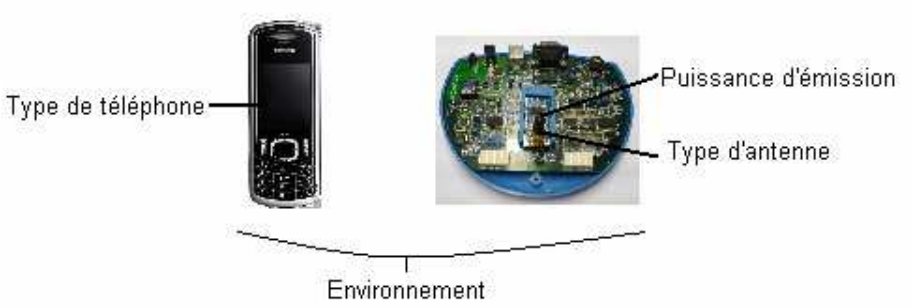


Figure 2 : Paramètres variables

2.1. Répétitivité de la mesure

L'accès aux fonctions de Bluetooth tel que l'activation ou la recherche d'appareil à proximité est différent d'un téléphone portable à l'autre. Pour obtenir une méthode de mesure répétitive et indépendante du type de téléphone portable, il est nécessaire de réaliser une application générique.

Il existe actuellement deux façons de programmer un téléphone portable :
- Java (J2ME)
- Symbian C++

Sous Symbian, le développement d'application passe en général par l'utilisation d'une API propriétaire souvent écrite en C ou C++. Le portage d'une application implique donc l'adaptation du code pour presque chaque modèle de téléphone …
Avec Java, nous pouvons entrevoir une solution standard pour tout type de téléphone. Sun nous propose une version allégée de J2SE, adaptée aux appareils aux ressources limitées. Cette plateforme de développement Java est appelée J2ME.

2.2. Programmation du téléphone portable

En programmant avec les librairies mises à disposition par J2ME le système est très portatif et peu fonctionner de manière équivalente sur la plupart des téléphones portables récents. Ceci est défini par la norme MIDP (annexe D.4).

Deux classes Java (Midlet et Canvas) sont disponibles pour gérer l'affichage du téléphone portable. Les principales différences sont décrites en annexe D.2. La classe Midlet est une API de haut niveau qui permet simplement d'accéder à l'écran du téléphone portable en y insérant des éléments prédéfinis (formulaire, zone de texte, alerte,…).

La JSR82 (annexe D.1) est l'API java qui met à disposition les fonctionnalités nécessaires pour accéder au Bluetooth d'un téléphone portable. Il est donc obligatoire que le téléphone portable la supporte.

Le téléphone portable doit être compatible également avec deux éléments de configuration :

- MIDP 2.0 est un ensemble d'API Java (annexe D.4).
- CLDC 1.1 est une configuration adaptée aux appareils aux ressources limitées tel que les téléphones portables. (Annexe D.3)

Les éléments nécessaires décrits ci-dessus peuvent être mis en forme à l'aide d'un schéma en couches :

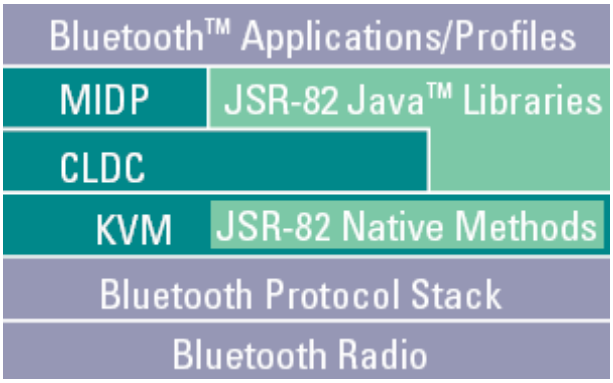
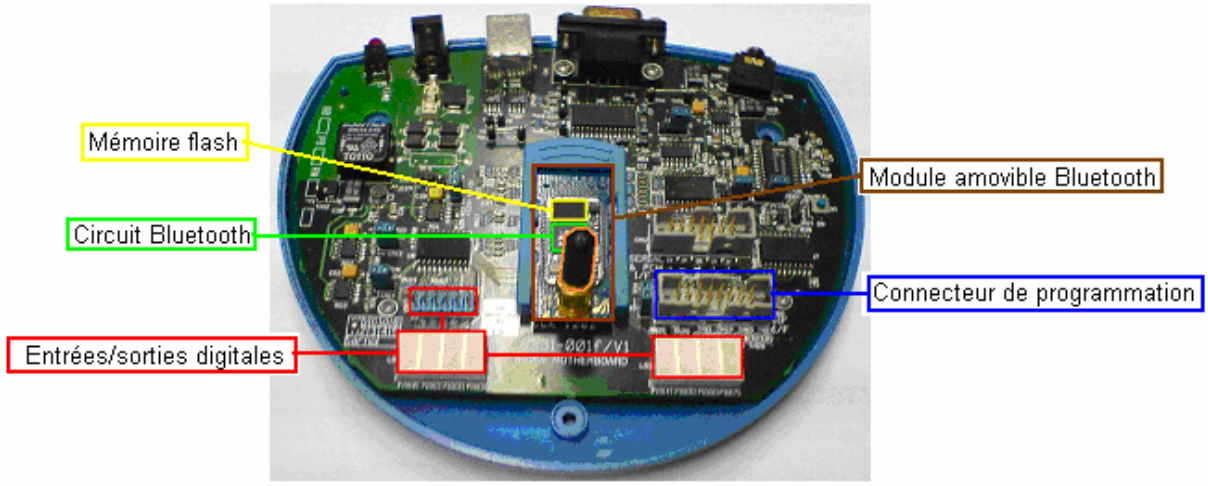


Figure 3 : Couches Bluetooth (Java)

KVM est la machine virtuelle Java destinée aux appareils aux ressources limitées. Le principe de la machine virtuel Java est de fournir le même environnement de travail sur toutes les plateformes (dans ce projet, sur tous les téléphones portables).

Actuellement tous les téléphones portables ne supportent pas la JSR82. Actuellement plus de soixante modèles de téléphone portable sont compatibles. La liste complète des modèles disponible se trouve en annexe C.

2.3. Programmation du Casira



Equipé d'un processeur ARM7 de type BlueCore 2, le système de développement Casira (annexe B.1) permet d'utiliser Bluetooth en ayant accès à des fonctions de bas niveau. Le BlueCore 2 est compatible Bluetooth v1.1 et 1.2.

Grâce aux librairies mises à disposition par le système de développement, il est notamment possible de paramétrer la puissance à laquelle le signal Bluetooth est émis. La modification de cette puissance permet d'augmenter ou de réduire la portée du Casira. Les détails des paramètres liés à la puissance sont accessibles en annexe B.1.2.

2.4. Etablissement d'une connexion Bluetooth

Il y a deux possibilités d'initier une connexion Bluetooth :

- le téléphone portable effectue une requête de connexion vers le Casira
- le Casira effectue une requête de connexion vers le téléphone portable

Lors des mesures, la première méthode est utilisée pour permettre à l'interface utilisateur de choisir quand la connexion s'établit. La seconde méthode implique une programmation plus complexe. Elle est expliquée et utilisée plus loin dans le projet (3. Objet intelligent).

2.4.1. Préalable

Avant de pouvoir établir la connexion entre le téléphone portable et le Casira, ils doivent être appariés car la connexion est réalisée à l'aide du profile SPP (annexe A.3.1). L'appariement est nécessaire en utilisant ce profil. Lors de cette opération, le Casira sauve l'adresse Bluetooth du téléphone portable et ne permettra qu'à ce seul téléphone de réaliser les mesures.

2.4.2. Connexion

La procédure de connexion liée à la première méthode est décrite ci-dessous.

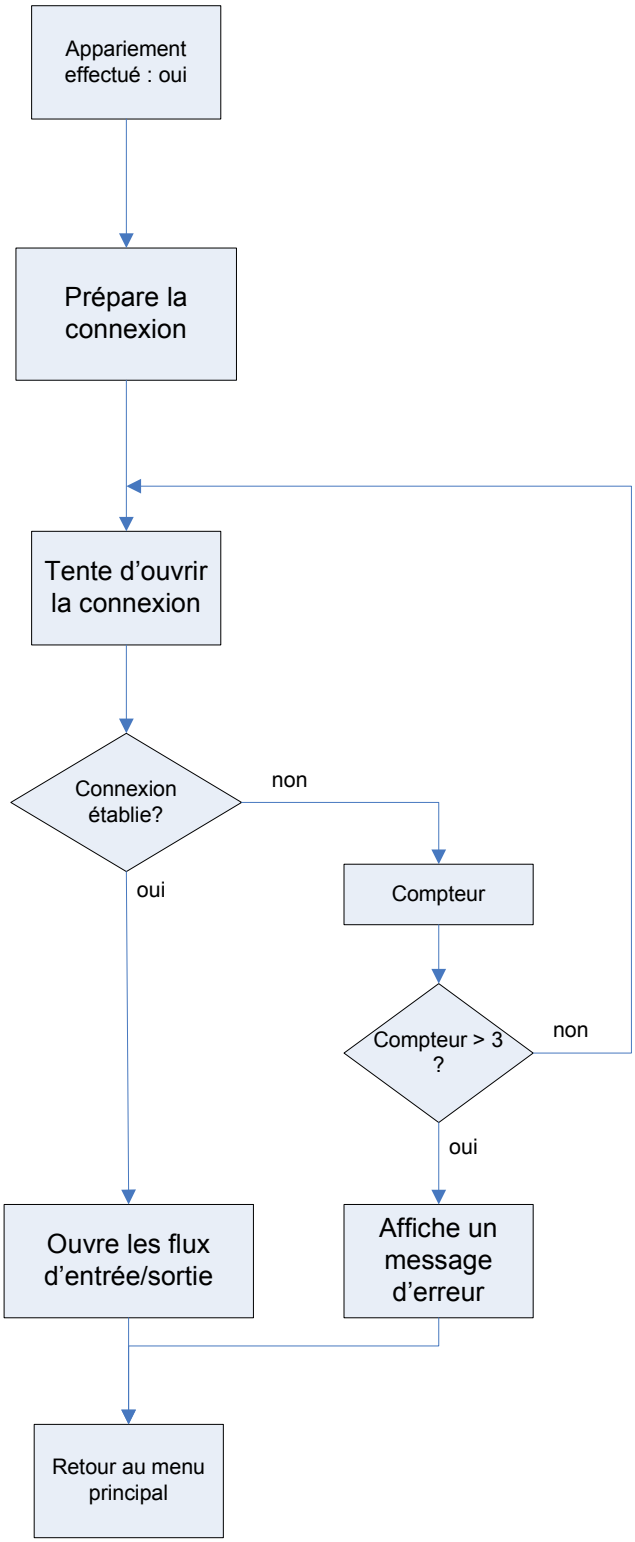


Figure 4 : Connexion client Bluetooth

Le code Java qui découle de cet organigramme est disponible en annexe I.2.

Lorsque la connexion est établie, j'utilise un protocole de message bien défini pour échanger des informations entre le téléphone portable et l'objet intelligent.

2.5. Protocole de communication

Les librairies mises à disposition par la JSR82 ne permettent pas d'accéder aux paramètres Bluetooth bas niveau du téléphone portable. Le seul moyen de limiter la portée à laquelle le téléphone portable et le Casira peuvent communiquer est de limiter la puissance d'émission du Casira.

Le Casira permet d'accéder, par programmation, à tous les paramètres Bluetooth mais n'offre pas d'interface utilisateur. J'utilise le téléphone comme interface pour accéder en lecture/écriture aux registres du Casira. Ceci est fait par un protocole de messages définis à l'annexe E.

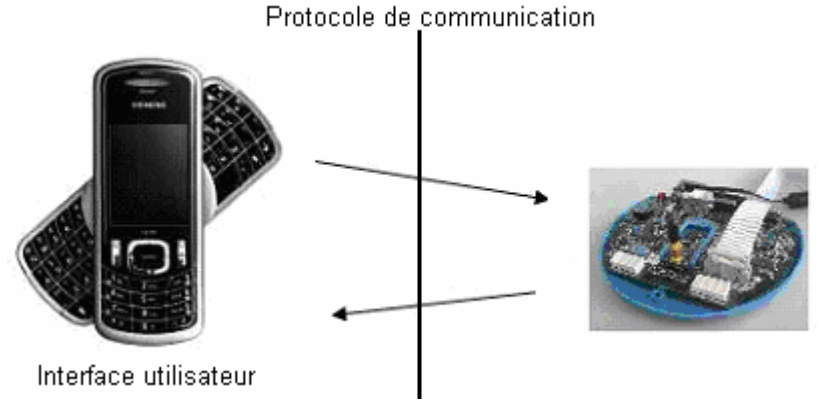


Figure 5 : Protocole de communication

2.6. Mesure de portée

Toutes les fonctionnalités de l'application java créée sont disponibles en annexe F. En plus de crée une interface permettant de modifier la puissance du signal émis par le Casira, l'application est capable de déterminer quand la connexion est perdue.

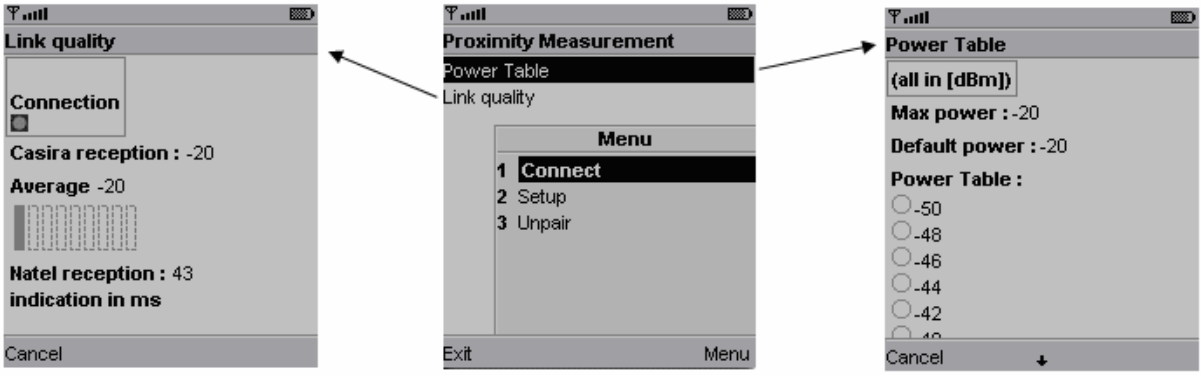


Figure 6 : Fonctionnalités du programme de mesure

Le menu permet d'enregistrer avec quel adresse Bluetooth il faut se connecter pour effectuer les mesures. Seul le champ « Connection » du formulaire « Link quality » a servit pour les mesures. Il détermine si les deux appareils sont connectés ou non. Les autres champs sont uniquement à titre indicatif. Ils estiment la qualité du signal échangé entre les deux appareils. Les valeurs de puissance disponible dans le Casira sont affichées sous forme de tableau. Il est possible de modifier la puissance facilement.

2.6.1. Paramètres possibles

Les différents paramètres qui vont être testés lors de la mesure sont :

- Le type de téléphone portable
- Le type d'antenne
- La puissance du Casira
- L'environnement de mesure

2.6.1.1. Type de téléphone

Chaque construction de téléphone étant unique, la position de l'antenne est également différente d'un téléphone à l'autre. Si l'antenne se trouve à l'endroit où le téléphone est pris en main ou à un lieu dégagé, l'influence sur la mesure sera certainement importante.

La sensibilité de l'antenne est également un facteur important. Une bonne antenne permet sans doute une détection du signal Bluetooth à une plus faible puissance d'émission qu'une antenne de qualité moindre.

2.6.1.2. Type d'antenne

Plusieurs géométries d'antennes (Annexe B.1.3) vont être testées pour évaluer leur influence sur la portée du Casira.

2.6.1.3. Puissance du Casira

En modifiant la puissance maximale fournie à l'antenne, il est possible de limiter la portée du Casira. Je vais réaliser diverses mesures pour établir quelle est l'influence de ce paramètre sur la portée réelle.

2.6.2. Environnement de mesure

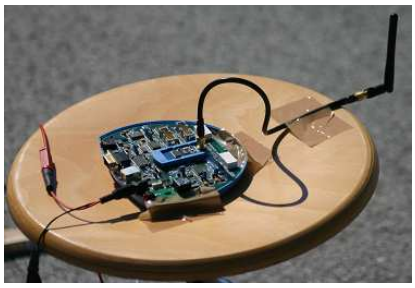
J'ai réalisé plusieurs essais de mesures aux abords de la HEvs à Sion. Ceux-ci m'ont rapidement montré que la place dégagée est trop petite pour atteindre la limite de la portée du téléphone portable. Pour modifier des paramètres tel que le type de téléphone portable ou le type d'antenne utilisée, il est nécessaire d'aller dans un environnement dégagé. L'objectif est que celui-ci n'influence pas le résultat de mesure. La transmission du signal radio Bluetooth est extrêmement influencée par son environnement.

L'aéroport désaffecté de Tourtemagne (Vs) est un terrain plus propice pour effectuer mes mesures. Il est possible d'y trouver une ligne droite de plus de 100m complètement dégagée. C'est donc sur ce terrain que je vais effectuer mes mesures.

2.6.2.1. Casira

Le Casira est fixé au centre d'un tabouret à une hauteur de 0.85m du sol. Il est alimenté par deux batteries branchées en parallèle prévues pour une autonomie de plus de 10h.

Deux types d'antenne sont utilisés. Soit l'antenne est placée directement sur le Casira, soit elle est reliée par un câble.



2.6.2.2. Téléphone portable

Le téléphone portable est tenu dans la main droite identiquement à une utilisation normale. Il est incliné d'environ 25°. La partie basse du téléphone est partiellement recouverte par la main.

Cette manière de faire à une influence assez importante selon la position de l'antenne dans le téléphone. Nous avons découvert qu'un type de téléphone portable avait son antenne à un endroit complètement recouvert par la main lorsqu'on tient le téléphone de manière correcte. Ce téléphone n'a pas été mesuré.

2.6.3. Déroulement des mesures

Les mesures ont été effectuées par beau temps sur une période de 9h à 14h. Elles ont tout été prises entre l'axe de l'antenne du Casira et le centre du téléphone portable.

Les mesures se déroulent en 4 étapes :

1. Paramétrage de la puissance du Casira
2. Connexion avec le Casira
3. Je recule jusqu'à ce que la connexion soit perdue.
4. La distance est notée

Chaque mesure est réalisée trois fois.

2.6.4. Extrait des résultats obtenus

Mesures obtenues avec trois antennes différentes :
Siemens K750i

N° d'antenne	Position de l'antenne	Puissance du Casira [dBm]	Portée moyenne mesurée[m]
2	vertical	-30	2.3
3	vertical	-30	3.7
4	horizontal	-30	1.7

On remarque que trois antennes différentes produisent trois mesures différentes.

Deux mesures très espacées dans le temps ont produit deux résultats voisins.
Siemens SK65

Heure approximative	N° d'antenne	Position de l'antenne	Puissance du Casira [dBm]	Portée moyenne mesurée[m]
09:30	2	vertical	6	126.3
13:20	3	vertical	6	127.5

On remarque également que les antennes 2 et 3 sont de qualité similaire pour la distance mesurée.

Mesure obtenue avec l'antenne 2 :

Puissance du Casira [dBm]	SK65 [m]	K750i [m]
-40	2.5	0.36
-36	5.1	0.7
-34	7.9	0.8
-32	8.3	2.0
-30	9.1	2.3

On remarque facilement que les deux téléphones produisent des résultats extrêmement différents. La différence est un peu moins grande lors de test de plus grande portée.

Les antennes correspondant aux numéros se trouvent en annexe B.1.3.
Le tableau complet des mesures effectuées est disponible en annexe G.

2.7. Résultats et théorie

Dans le cas d'une liaison en espace libre, si la distance *d* entre l'émetteur et le récepteur augmente la surface apparente de l'antenne de réception diminue, ce qui se traduit par un accroissement de l'atténuation. En ligne de vue et dans un environnement dégagé, l'affaiblissement, appelé affaiblissement de parcours est proportionnel au carré de la distance d^2. Selon le type d'environnement, l'affaiblissement est proportionnel à une puissance supérieure de *d*. Par exemple pour un environnement urbain, l'affaiblissement est estimé à $d^{3.5}$.

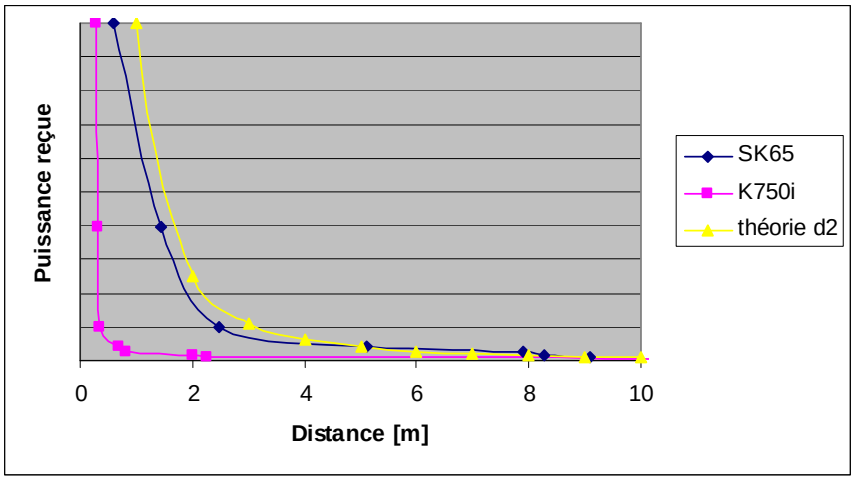


Figure 7 : Résultat des mesures

L'établissement d'une courbe véritable nécessite d'exécuter les mesures un grand nombre de fois. Pour ne pas que les mesures ne me prennent trop de temps, chaque mesure a été réalisée trois fois excepté pour les puissances les plus faibles. Ceci m'a permis d'obtenir une tendance de la courbe. Il faudrait réaliser plus de mesures, notamment pour les distances au dessous de 2m, c'est là ou se situe le coude du graphique.
La courbe jaune correspond à la valeur théorique obtenue avec un affaiblissement proportionnel au carré de la distance d^2. Elle est placée sur le graphique en considérant une portée de 10m pour une puissance de 1mW (définit par la Classe III Bluetooth). Le fait que les résultats soient assez proches de la théorie dans un environnement dégagé montre la pertinence de la mesure.

2.8. Conclusion

Le principal enseignement à retenir de ces mesures est qu'il n'est pas possible d'établir une correspondance directe entre la puissance paramétrée dans le Casira et la portée réelle perçue par un téléphone portable.

Etant donné que Bluetooth travaille à des fréquences extrêmement élevées (2.4Ghz), les signaux transmis sont très sensibles à tout élément qui se trouve aux environs de la connexion Bluetooth.

Il est donc obligatoire de passer par une phase d'étalonnage pour faire cette correspondance. L'étalonnage est valable uniquement pour un type d'antenne associé à un type de téléphone dans un environnement donné.

3. Objet intelligent

3.1. Caractéristiques de l'objet intelligent

L'objectif est d'implémenter une liste de propriétés et de comportements disponibles dans l'objet concerné. Lorsqu'un utilisateur s'approche de l'objet intelligent, une connexion est établie et les caractéristiques de l'objet sont envoyées sous forme de liste à un téléphone portable connu et à proximité. Elle est affichée sur l'écran du téléphone portable et crée une interface avec l'objet intelligent.

L'objet intelligent est défini d'après une liste de critères :

- Le type d'objet (décrit en annexe A.7)
- Le nom
- Les propriétés
- Les comportements
- Les types des comportements
- Les actions à effectuer en fonction des commandes reçues et des entrées

La déclaration détaillée des différents champs est disponible en annexe J.1.

L'objet intelligent est déclaré en une liste de propriétés et de comportements :

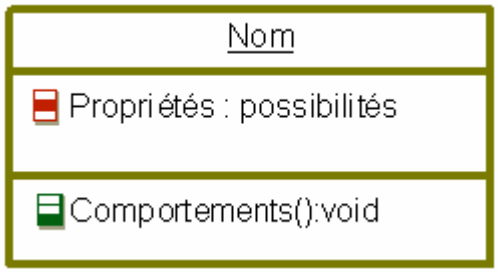


Figure 8 : Déclaration de l'objet intelligent

L'objet intelligent est programmé de manière extrêmement adaptative au contexte. En créant une application le plus générique possible, plusieurs possibilités s'offrent pour modifier les caractéristiques de l'objet (point 6.Améliorations à venir).

3.2. Qui initie une connexion Bluetooth ?

L'objet intelligent recherche de manière continue si un téléphone portable apparié se trouve dans son rayon d'émission. C'est l'objet intelligent qui est l'initiateur de la connexion. D'après la notion de Client/Serveur Bluetooth décrite en annexe A.6, l'initiateur de la connexion est client. Il se connecte à un service mis à disposition par un serveur Bluetooth.

Il faut donc créer un serveur Bluetooth en Java pour permettre à l'objet intelligent de se connecter au téléphone portable. La création du serveur à pour objectif de mettre à disposition un service propriétaire nommé « Remote Control ».

Les différentes étapes nécessaires à la création d'un serveur Bluetooth sont décrites en annexe I.3.

Profile SPP

Si le Casira initie la connexion avec le SPP '' standard '', la connexion est établie entre les deux appareils, mais elle n'est pas détectée par le Java. L'utilisation du protocole de message est par conséquent impossible.

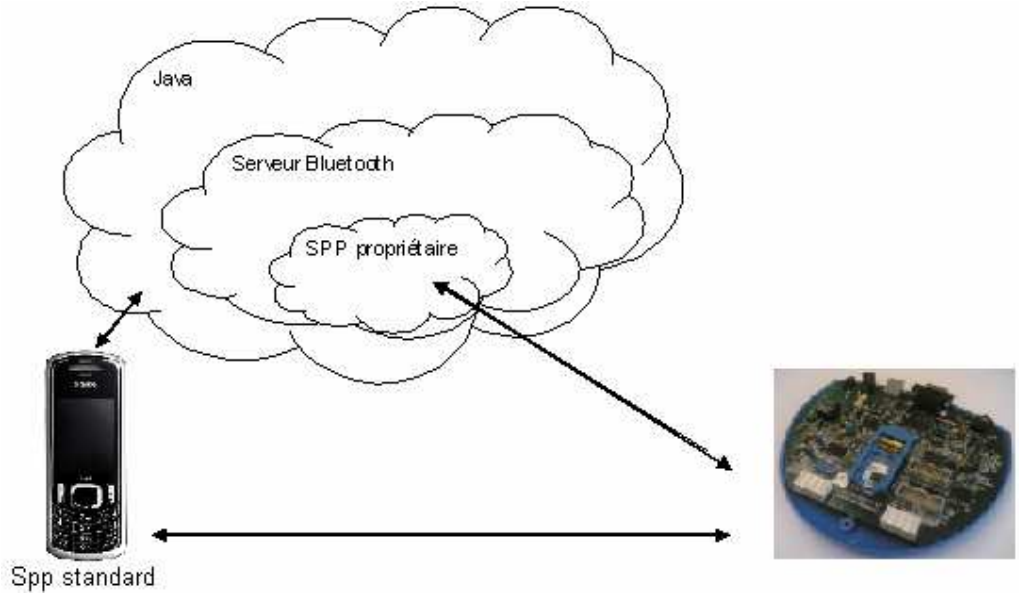


Figure 9 : Spp '' standard '' et propriétaire

L'implémentation d'un serveur Java nécessite donc de déclarer un service propriétaire pour permettre l'acceptation de la connexion par le Java. La déclaration de notre propre SPP nous ouvre des portes et permet de faire fonctionner l'application sur des téléphones portables qui ne supportent pas le SPP standard.

Etant donné qu'on utilise un SPP, il est nécessaire que les deux appareils soient appariés. Plus de détails sur les profils Bluetooth sont disponibles en annexe A.3.

3.3. Demande de connexion par l'objet intelligent

Casira

Il n'est pas possible pour l'objet intelligent d'être découvert lorsque celui-ci est à la recherche d'un téléphone portable apparié. Ceci vient du fait que l'objet intelligent consacre entièrement son temps à la recherche et ne répond pas (ou très rarement) à un téléphone portable qui essaie de s'apparier avec lui.

Un bouton de configuration qui permet de choisir lorsque l'appareil peut être apparié ou non à été mis en place. Les deux choix offerts sont :

- le mode « appariable »
- le mode « recherche »

Le mode « appariable » permet à un téléphone portable de découvrir l'objet intelligent et d'effectuer le processus d'appariement nécessaire avant de pouvoir établir une connexion entre les deux appareils. Cette procédure est obligatoire dans le cas d'une connexion à l'aide d'un profil SPP.

Le mode « recherche » indique que le Casira scrute de manière continue si les téléphones portables appariés sont à portée et exécutent l'application Java. La recherche est exécutée de manière spécifique sur le service « Remote Control » mis à disposition par le serveur Bluetooth Java.

Le profil SDP (annexe A.3.2.) permet de découvrir si un appareil Java met à disposition un service spécifique. C'est grâce à ce profil que cette tâche est réalisée.

Téléphone portable

Le téléphone portable doit activer Bluetooth et exécuter l'application java qui contient le Serveur Bluetooth. Certains téléphones portables permettent de réduire l'application. Ceci permet à l'utilisateur d'utiliser librement son téléphone. Il n'est toutefois pas possible de lancer une seconde application Java.

3.4. Résultats obtenus

Lorsqu'un téléphone portable connu et apparié se rapproche suffisamment de l'objet intelligent, ici l'objet « Garage » (Déclaré en 3.1, Figure 8), la connexion s'établit et produit le résultat ci-dessous :

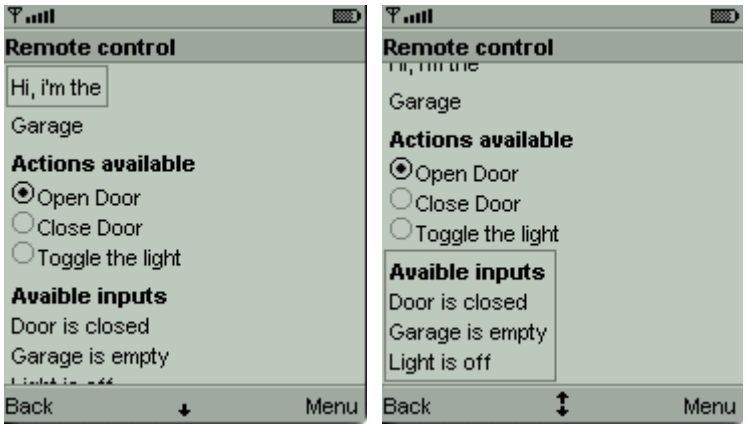


Figure 10 : « Garage » connecté

Un ascenseur permet de se déplacer verticalement dans le menu lorsque les propriétés et comportements disponibles sont trop nombreux.

Les commandes sont testées grâce aux leds disponibles sur le système Casira. Un interrupteur est ajouté pour simuler le changement d'un attribut. L'exécution des méthodes est fonctionnelle ainsi que la mise à jour de la valeur des attributs lorsque l'un d'entre eux est changé. Lorsque le téléphone sort de la zone d'émission de l'objet intelligent, le menu est désactivé et un nouveau formulaire apparaît à l'écran.

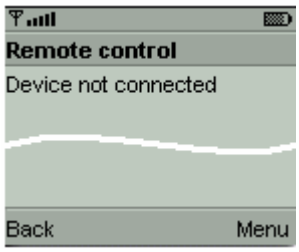


Figure 11 : « Garage » déconnecté

L'interface utilisateur permet de modifier le comportement de l'objet ainsi que d'en récupérer les propriétés.

Des tests de comportement ont été réalisés avec le Siemens SK65 et le Sony Ericsson k750i. A la connexion, le SK65 demande chaque fois une confirmation tandis que le k750i peut être configuré pour accepter la connexion sans la demande. L'affichage est quelque peut différent mais la fonctionnalité est complètement identique. Tests complets en annexe H.

4. Maquette de démonstration

Une maquette de démonstration permet de montrer à quel point l'étude de proximité et la programmation développée au préalable peuvent être adaptés à des objets de la vie courante. Il peut également expérimenter la notion de limite d'accessibilité d'un objet intelligent à un périmètre restreint.

Figure 12 : Maquette de démonstration
La télévision ci-dessus n'apparaît pas comme dans la réalité.

Deux objets intelligents sont implémentés sur la maquette :

- Le garage
- La télévision

L'ouverture et la fermeture de la porte du garage est possible grâce à un moteur relié à celle-ci. La lampe adjacente au garage peut également être commandée. Le moteur et la lampe sont tous deux reliés à une carte de commandes (annexe B.6) qui permet de fournir la tension nécessaire à leurs bon fonctionnements.

Un PDA déguisé en télévision est connecté par Bluetooth à la carte de commandes (annexe B.6). Une application a été développée au sein de l'unité infotronics pour communiquer avec la carte de commandes et afficher des images.

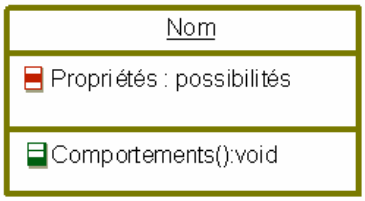
Le garage est accessible uniquement quand on se trouve à une certaine distance de sa porte. Lorsqu'on rentre dans la maison, il devient inaccessible. En se dirigeant dans le salon, la télévision devient accessible et le menu apparaît grâce à la même application Java exécutée dans le téléphone portable.

Il est tout à fait possible d'ajouter à ce démonstrateur d'autres objets intelligents :

- La salle de bain (objet qui comporte le chauffage, la lampe, la porte)
- La machine à café
- Une chaîne hi-fi
- tout objet électrique qui nécessite une commande

4.1. Les objets intelligents

Comme vu précédemment, les objets intelligents sont déclarés de cette manière :



Le garage

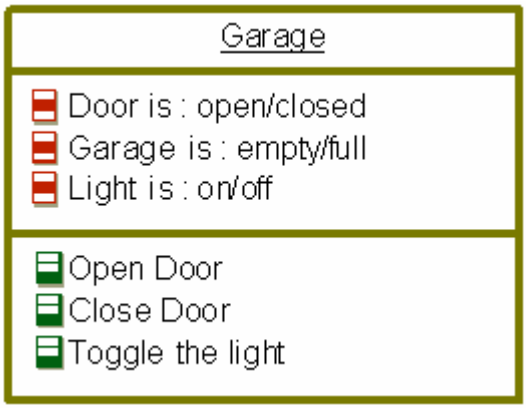


Figure 13 : Déclaration de l'objet garage

Il est possible d'influencer le comportement de l'objet de différentes manières :

- Ouverture de la porte
- Fermeture de la porte
- Inversion de l'état de la lampe.

L'objet met à disposition les propriétés suivantes :

- Etat de la porte (ouverte/fermée)
- Etat du garage (occupé/ libre)
- Etat de la lampe (allumée/éteinte)

L'objet garage ne contient ni de fin de course relié à la porte, ni d'électronique permettant d'établir si la lampe est allumée. Je simule ces informations dans la programmation de l'objet pour que les propriétés de l'objet soient mise à jour correctement.

Un capteur à effet hall placé dans le garage permet de modifier la propriété « Etat du garage ». Celui-ci détecte si la voiture miniature munie d'un aimant se trouve dans le garage.

La télévision

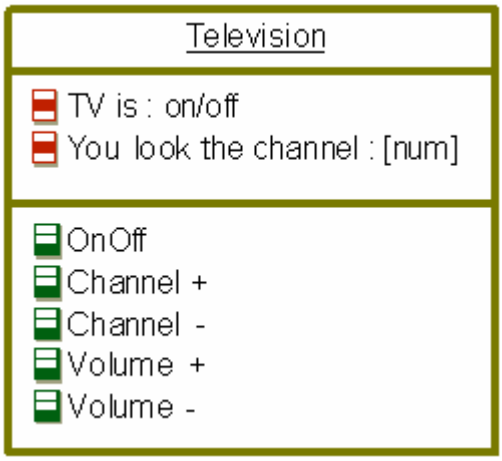


Figure 14 : Déclaration de l'objet Télévision

Il est possible d'influencer le comportement de l'objet de différentes manières :

- Inversion de l'état de la télévision
- Augmentation d'un canal
- Diminution d'un canal
- Augmentation du volume
- Diminution du volume

L'objet met à disposition les propriétés suivantes :

- Etat de la télévision (allumée/éteinte)
- Chaîne regardée (1…6)

L'objet télévision réalisé ne renvoie pas d'information à la carte de commandes. Ces informations sont simulées dans la programmation de l'objet pour que les propriétés de l'objet soient mise à jour correctement.

4.2. Fonctionnalités de la maquette

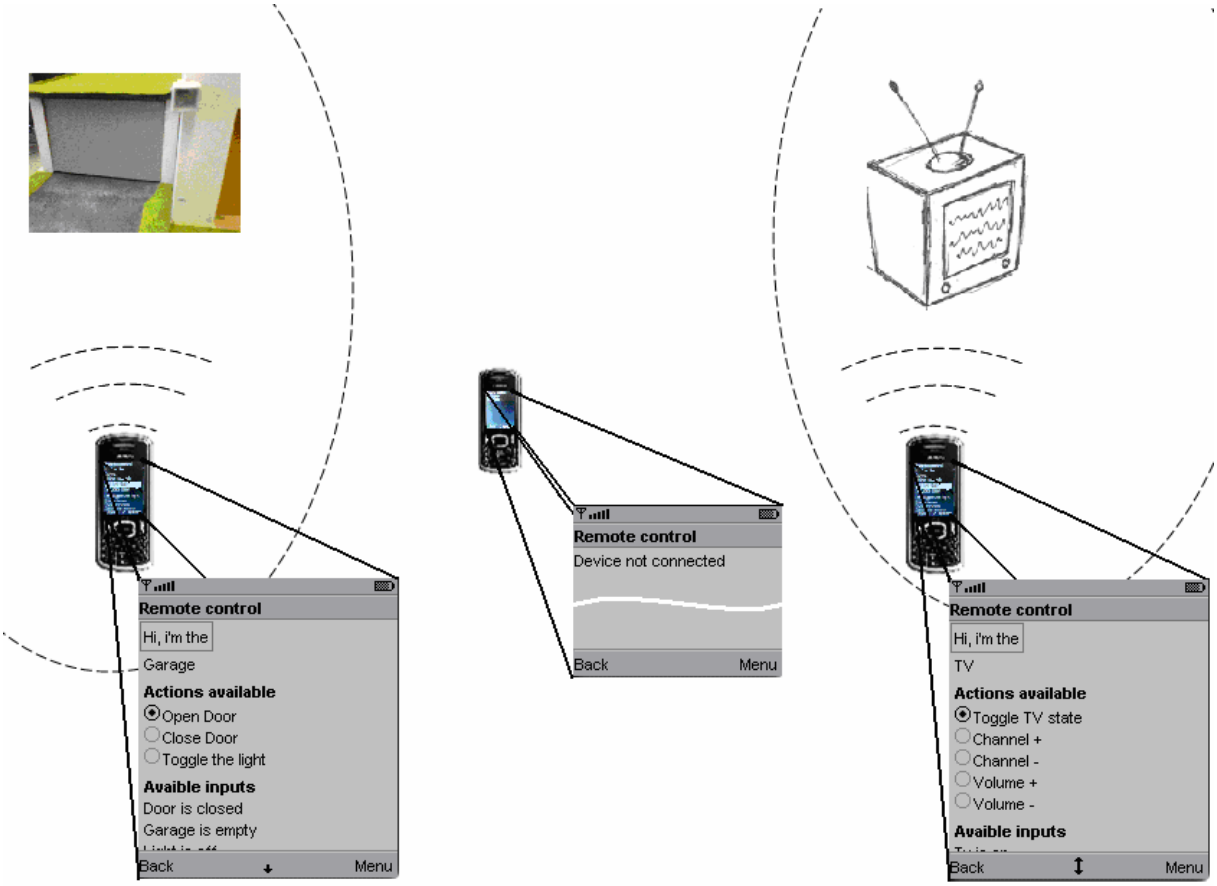


Figure 15 : Fonctionnalités de la maquette

La maquette permet de démontrer l'adaptabilité de mes études à un environnement domestique. Elle permet de montrer la faisabilité de limiter la portée d'émission d'un objet intelligent. On remarque également la souplesse de l'application Java. Celle-ci permet d'afficher les propriétés et comportements d'un objet intelligent de façon adaptative au contexte.

5. Conclusion du projet

Les mesures ont montré à quel point la distance (pour une même portée paramétrée dans l'objet intelligent) à laquelle la connexion est disponible dépend du téléphone portable. Les résultats obtenus diffèrent d'un coefficient deux à trois en modifiant uniquement le téléphone portable. De plus, les mesures n'ont été réalisées qu'avec deux types de téléphone portable dans le même environnement.

L'environnement dans lequel le système est utilisé influence également fortement la portée. Par exemple lorsqu'on utilise l'objet intelligent « Garage », il n'est pas possible d'établir la distance maximale de la connexion, cela dépend de trop de paramètres :

- la géométrie du garage ou de la maison contre lequel ou laquelle l'objet intelligent est fixé

- le type de voiture dans laquelle on se trouve
    - la situation du téléphone dans la voiture :
        - est-ce qu'il est sur le siège
        - dans la poche
        - dans la veste
            - type de veste, épaisseur du tissu
            -…

Tous ces paramètres obligent l'utilisateur à effectuer un étalonnage du système avant son utilisation.

La maquette permet de montrer l'avancement du projet, en particulier :

- la communication entre les différents appareils.
- qu'il est possible de gérer la distance de connexion

Par contre, elle ne montre pas la nécessité de passer par un étalonnage ainsi que l'influence de l'environnement dans lequel le système est implanté.

Dans l'objectif d'un développement du projet pour le commerce, il reste principalement deux points à améliorer :

- L'utilisation des Canvas est obligatoire pour obtenir une application élégante et complètement identique sur tous les téléphones portables.
- La réactivité du système est à étudier plus en profondeur. Avec un délai de vingt secondes avant que la connexion soit perdue, il est sûr que l'utilisateur va être irrité lorsqu'il essaie d'envoyer les commandes alors que la connexion est perdue, mais n'est pas encore détectée comme telle.

6. Améliorations à venir

Distribution du programme Java

Actuellement je transfère le programme Java de mon ordinateur au téléphone portable soit par un câble disponible avec celui-ci, soit avec un dongle Bluetooth. Pour une utilisation grand public du système, il existe une solution plus pratique.
Il est possible d'enregistrer le programme java dans la mémoire flash du système Casira et de le faire parvenir au téléphone portable lorsque les deux appareils sont appariés. Cette fonctionnalité a été testée aux seins de l'unité infotronics mais n'a pas été implémentée dans mon projet.

Possibilités des Canvas

Le résultat obtenu en programmant avec les Canvas plutôt que les Midlets (Détails en annexe D.2) est beaucoup plus élégant.

En utilisant les Canvas, le programme Java reste majoritairement identique à l'actuel. L'échange de messages reste identique en grande partie, mais il n'est pas exclu que celui-ci nécessite une légère adaptation. Ce qui concerne l'affichage à l'écran est entièrement à revoir.

Les fonctions mises à disposition par l'objet intelligent « télévision » pourraient produire ce résultat sur le téléphone portable :

Figure 16 : Exemple de Canvas

Le choix de la commande peut être effectué de deux manières :
- en déplaçant le curseur rouge
- en utilisant les touches numériques du téléphone

En utilisant les Canvas, il est possible d'utiliser les touches numériques du téléphone pour exécuter les commandes. On obtient ainsi une télécommande qui permet d'accéder à un grand nombre de fonctions facilement.

Mise à jour de l'objet intelligent

On peut imaginer que les propriétés d'un objet intelligent soient modifiées.
Par exemple :

- Ajout d'une lampe dans le garage qui est reliée à une entrée de l'objet intelligent
- Ajout d'un élément supplémentaire à une chaîne stéréo

Ceci nécessite une mise à jour de l'intelligence de l'objet. Différentes manières peuvent être utilisées pour réaliser cette tâche :

- par un port USB
- par Bluetooth

L'ajout d'un port USB à l'objet intelligent permet de modifier sa configuration à l'aide d'une clé USB qui contient la nouvelle configuration de l'objet obtenu soit:

- par le fabricant de l'objet.
- téléchargée depuis internet

L'utilisation de Bluetooth pour reconfigurer un objet intelligent offres également plusieurs possibilités. La configuration est :

- envoyée par le réseau de téléphonie mobile à l'utilisateur puis à l'objet intelligent
- téléchargée depuis internet et installée grâce à un dongle Bluetooth (ou grâce au téléphone)

Un risque cependant se manifeste. Il est identique à celui qui apparaît aujourd'hui avec un grand nombre d'applications qui tournent sur nos ordinateurs. Il est possible qu'un constructeur développe l'intelligence de l'objet sans effectuer tous les tests avant sa commercialisation. Il publie ensuite des correctifs à cause de cette possibilité offerte : la mise à jour de la configuration.

Réactivité du système

Principalement deux opérations prennent du temps :

- Lors de la recherche par l'objet intelligent si un téléphone est à portée. Le délai d'attente avant de rechercher le téléphone suivant est de 5sec. Le temps de cycle dépend donc du nombre de téléphones appariés.

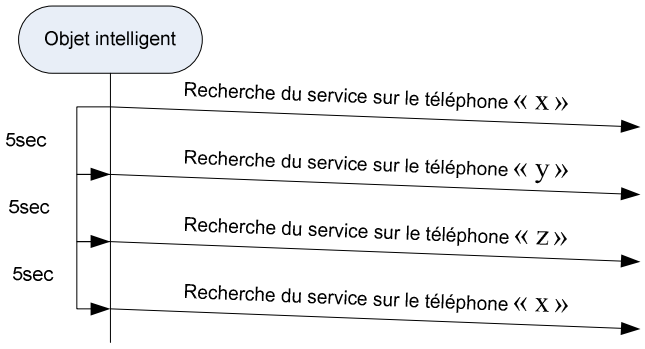


Figure 17 : Réactivité du système

- L'opération qui prend le plus de temps, est la détection de la perte de connexion lorsque les deux appareils ne sont plus à portée. Actuellement, ce temps est de 20sec. Ce délai n'est pas admissible pour une utilisation grand public du système. Il peut être réduit en créant un système de « ping » entre les deux appareils, il est possible de détecter quand la connexion est perdue de manière plus réactive.

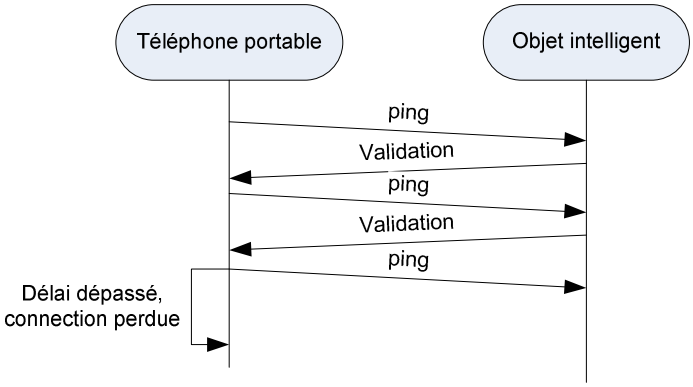


Figure 18 : Implémentation d'un ping

Le choix du ''délai dépassé'' est cette fois possible grâce à l'application Java. En fixant ce délai à une seconde par exemple, on obtient un système beaucoup plus réactif.

Multiplication des objets intelligents

Lorsque deux objets intelligents apparaissent dans le même environnement, il est possible que l'utilisateur ait envie de les commander tous les deux. (Par ex. le son d'une télévision reliée à une chaîne hi-fi) La solution actuelle ne permet qu'une connexion unique. Si l'utilisateur se trouve dans un environnement contenant deux appareils, le premier d'entre eux à détecter le téléphone se connectera à celui-ci.

Deux possibilités sont possibles pour résoudre ce problème :

- Un menu supplémentaire apparaît lorsque deux connexions sont demandées
- Les deux listes de propriétés et comportements s'affiche à la suite

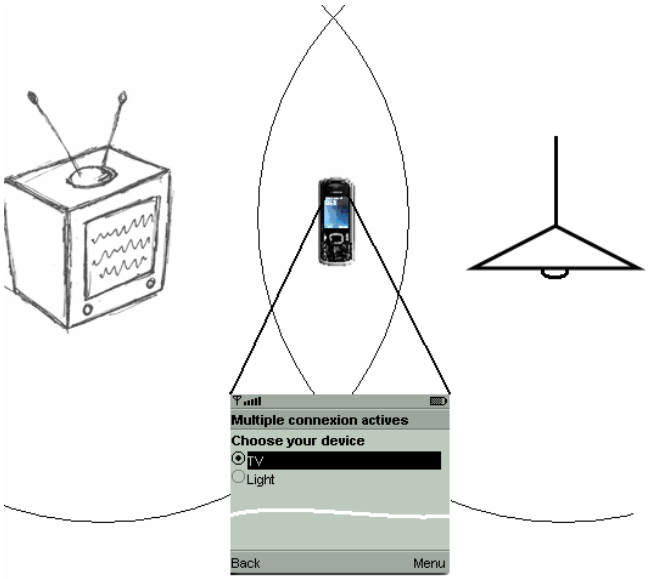


Figure 19 : Multiplication d'objets intelligents

Le menu supplémentaire me semble être une bonne solution dans le cas ou les deux objets intelligents ne sont pas liés (P.ex. Télévision et Lampe du salon). Dans le cas où ils sont intimement liés (P.ex. Télévision et chaîne stéréo), la solution de l'affichage des deux menus à la suite me parait meilleure. Un élément de configuration supplémentaire permettrait de déterminer quels objets sont liés. Celui-ci pourrait être accessible directement par l'objet intelligent si le lien entre les deux est de type Bluetooth.

7. Sources

- Patrice Rudaz Adjoint scientifique à la Hevs: Programme java, C ANSI et connaissances concernant Bluetooth
- Google.ch : Recherche débouchant sur des articles sans auteur défini
- bluetooth.org : informations spécifiques sur les services, UUID, attributs d'un profil
- http://developers.sun.com/techtopics/mobility/apis/articles/bluetoothcore/ : Création d'un serveur Bluetooth en Java
- http://jcp.org/en/jsr/all : Liste des JSR mise à disposition par Sun
- Wikipedia.com : mot de recherche : Bluetooth

8. Glossaire

API	Outil logiciel, constitué d'un ensemble de modules dotés de fonctions communes, qui permet de produire automatiquement des applications Java.
CLDC	Configuration de J2ME pour les systèmes aux ressources limitées (annexe D.3)
J2ME	Le Java 2 Microedition est une plateforme Java conçue pour fonctionner sur des terminaux embarqués.
Java	Java est une technologie composée d'un langage de programmation orienté objet et d'un environnement d'exécution.
JSR	Les JSR sont des demandes d'utilisateurs du langage Java pour ajouter ou modifier des fonctionnalités sur la plate-forme officielle fournie par Sun.
JSR82	Ensemble d'API standardisé qui permet le développement d'applications dans un environnement Bluetooth.
KVM	Machine virtuel Java conçue pour les systèmes aux ressources limitées
MIDP	Ensemble d'API J2ME qui définit la façon dont les applications de logiciel se connectent à l'interface des téléphones cellulaires.(annexe D.4)
PDA	
SDDB	Lieu où sont défini et enregistrés les profils mis à disposition par un appareil Bluetooth.
URL	Adresse qui permet la connexion entre deux appareils
UUID	Identifiant unique d'un profil Bluetooth.

9. Annexes

A. Généralités concernant Bluetooth

Au 10ème siècle, le roi du Danemark qui portait le nom d'Harald Blåtand (littéralement dent bleue) unifiait le Danemark et la Norvège dans une Europe divisée par des querelles de religions et de territoires. Le symbole fort de cette unification a inspiré l'entreprise scandinave Ericsson qui est à l'origine de la technologie sans fil Bluetooth.

Les bandes ISM sont des bandes de fréquences qui ne sont pas soumises à des règlementations nationales et qui peuvent être utilisées librement pour des applications Industrielles, Scientifiques et Médicales. La technologie sans fil Bluetooth, comme le Réseau GSM, émet dans cette gamme de fréquence (2.4Ghz). Une liaison radio traditionnelle est conçue pour transmettre et recevoir sur un seul canal (une seule fréquence) ; ceci rend la liaison vulnérable à l'interception et sensible aux perturbations. Pour des raisons de sécurité et de fiabilité, Bluetooth utilise le saut de fréquence ou « frequency hopping ». Cette particularité permet de conserver une liaison même si quelques canaux sont perturbés en continu. Il existe 79 canaux Bluetooth, d'une largeur de bande d'1MHz chacun. Bluetooth utilise également un système de cryptage pour accroître la sécurité de la transmission.

Le réseau Bluetooth est basé sur un système maître/esclave : un des appareil joue le rôle de maître et gère les sauts de fréquences. Les appareils qui se situent dans le rayon de portée sont capables de se détecter sans intervention de la part de l'utilisateur et se rassemblent en sous réseaux appelés piconets. Un piconet ne comporte que 8 appareils actifs au maximum mais plusieurs piconets adjacents peuvent interagir. Ce réseau de piconets s'appelle un scatternet.

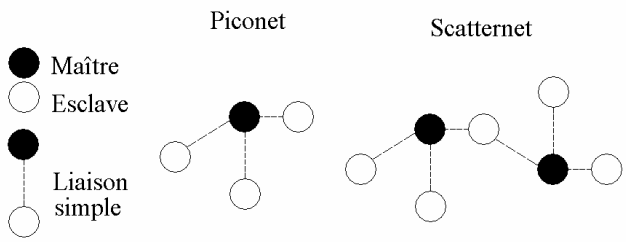


Figure 20 : Piconet et Scatternet

Le standard Bluetooth définit 3 classes d'émetteurs proposant des portées différentes en fonction de leur puissance d'émission. La plupart des fabricants d'appareils électroniques possédant cette technologie sans fil utilisent la classe 3.

Classe	Puissance	Portée
I	100 mW	100m
II	2.5 mW	15-20m
III	1 mW	10m

Les puissances ci-dessus sont souvent la source de questions concernant la nocivité des ondes émises. Les ondes Bluetooth ont un taux de pénétration de 1,5 cm à l'intérieur du corps. A

titre de comparaison, elle est de 2,5 cm pour un téléphone cellulaire fonctionnant de 10mW à 2 W à la fréquence GSM (900 Mhz). L'exposition au rayonnement émis par un téléphone cellulaire augmente la température de l'ordre de 0,1 °C ce qui laisse supposer une valeur négligeable pour la technologie Bluetooth.

A.1. Etablissement d'une connexion

La première étape de la communication Bluetooth est la recherche de périphériques disposant de la technologie dans notre rayon d'émission. Lorsque le second appareil est découvert, il faut échanger le mot de passe afin de devenir apparié (selon le type de connexion). Cette procédure permet d'effectuer une connexion temporaire ou définitive. Les paramètres tels que l'adresse Bluetooth et le port de communication sont enregistrés par les deux appareils. Ceci permet une connexion plus rapide lors d'une prochaine communication (évite de passer par la recherche relativement longue).

A.2. Les différentes couches Bluetooth

Afin d'assurer une compatibilité entre tous les périphériques Bluetooth, la majeure partie de la pile de protocoles est définie dans la spécification.

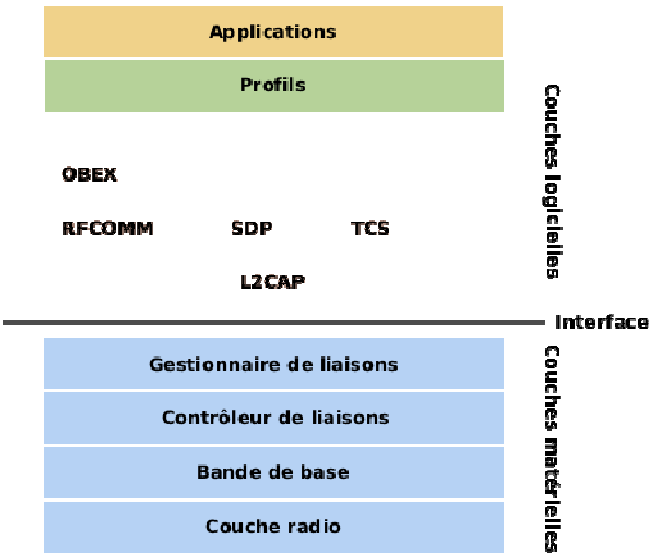


Figure 21 : Couches Bluetooth

A.2.1. Couches matérielles

La **couche radio** s'occupe de l'émission et de la réception des ondes radio. Elle définit les caractéristiques du transmetteur et du récepteur.

La **bande de base** est la couche physique du Bluetooth. Elle contrôle les canaux et les liens physiques indépendamment d'autres fonctions comme la correction d'erreurs, les sauts de

fréquence et la sécurité de Bluetooth. C'est également elle qui définit l'adresse matérielle unique de chaque périphérique.

Le **contrôleur de liaisons** (LC) a comme rôle de traiter les fonctions manipulées par la bande de base. Il exécute également le logiciel qui implémente le gestionnaire de liaisons.

Le **gestionnaire de liaisons** (LM) régit la communication entre les appareils Bluetooth. Ceci inclut notamment l'établissement de la connexion et l'authentification. Il découvre d'autres gestionnaires de liaisons à distance et communique avec eux. Pour exécuter son rôle de fournisseur de service, le gestionnaire de liaisons emploie les services du contrôleur de liaisons.

A.2.2. Couches logicielles

La couche **L2CAP** a pour rôle de transformer les données en paquets pour la couche Bande de base. Son rôle est également la gestion des connexions logiques entre plusieurs éléments Bluetooth.

RFCOMM est un protocole simple de transport qui fournit l'émulation des ports série RS-232.

Le protocole de découverte de service (**SDP**) fournit des moyens pour les applications de découvrir quels services sont disponibles et de déterminer les caractéristiques de ces services.

Le protocole de commande de téléphonie (**TCS**) définit la manière d'envoyer des appels audio entre les dispositifs de Bluetooth.

OBEX est un protocole employé couramment pour des transferts de fichier entre les dispositifs Bluetooth. Il permet par exemple le transfert de carte de visite, d'image ou encore de fichier musicaux.

A.3. Les profils

Le standard Bluetooth définit un certain nombre de profils d'application, permettant de définir le type de services offerts par un périphérique. Les profils ont pour but d'assurer une interopérabilité entre tous les appareils Bluetooth. Chaque périphérique peut supporter plusieurs profils.

Ils définissent :

- la manière d'implémenter un usage défini
- les protocoles spécifiques à utiliser
- les contraintes et les intervalles de valeurs de ces protocoles
- la manière dont deux périphériques communiquent entre eux pour une tâche définie
- si l'appariement est obligatoire pour se connecter

Voici quelques exemples de profils :

- File Transfer Profile (FTP) : profil de transfert de fichiers
- Dial-up Networking Profile (DUN ou DUNP) : profil d'accès réseau à distance
- Hands-Free Profile (HFP) : profil mains libres
- Serial Port Profile (SPP) : profil de port série

Le profil que je vais utiliser pour la communication entre le téléphone portable et le Casira est le profil SPP. Ceci vient du fait que la JSR82 qui permet d'accéder au Bluetooth du téléphone portable ne permet uniquement de travailler avec les profils OBEX et SPP.

A.3.1. Serial Port Profile (SPP)

Ce profil émule un câble série pour fournir un remplacement sans fil. Par conséquent, il est possible de faire transiter tous types de données grâce à ce profil.

Il est réalisable d'utiliser le profile SPP sur les téléphones portables qui ne le supportent pas en utilisant la JSR82 (ensemble d'API Java pour Bluetooth). Etant donné que la plupart des téléphones portables récents supportent le Java, il est tout à fait possible d'utiliser ce profile de manière extrêmement répandue.

Avant de réaliser une connexion à l'aide d'un profil de ce type, il est nécessaire d'apparier les deux appareils.

A.3.2. Service Discovery Protocol (SDP)

Le profil ''Service Discovery Protocol'' permet à un périphérique de demander à un appareil Bluetooth les profils disponible ainsi que leurs caractéristiques. Ce profil est utilisé sans appariement.

SDP dans l'application C

J'utilise ce profil pour déterminer si un téléphone portable apparié exécute l'application Java qui permet d'utiliser les fonctions de l'objet intelligent et d'en récupérer les attributs.

Deux méthodes de recherche sont possibles : la recherche s'effectue de manière série ou parallèle. L'inconvénient de la recherche série est le délai (5sec) avant que l'objet intelligent décide qu'un téléphone portable est hors de portée. La solution la plus rapide est la recherche parallèle.

Recherche du service propriétaire

La connexion avec un téléphone portable qui n'exécute pas de l'application Java ne doit pas être possible. Pour empêcher ceci, j'effectue une recherche sur le service mis à disposition par le serveur Java.

Effectuer la recherche sur tous les appareils appariés prends du temps et réduit la réactivité du système. Le temps de cycle qui est nécessaire pour tester à nouveau un téléphone portable dépend de la formule :

*t = Délai dépassé * Nombre de téléphones appariés*
(Délai dépassé apparaît ci-dessous vaut actuellement 5sec)

Une recherche des services sur chacun des téléphones portables produit le résultat suivant :

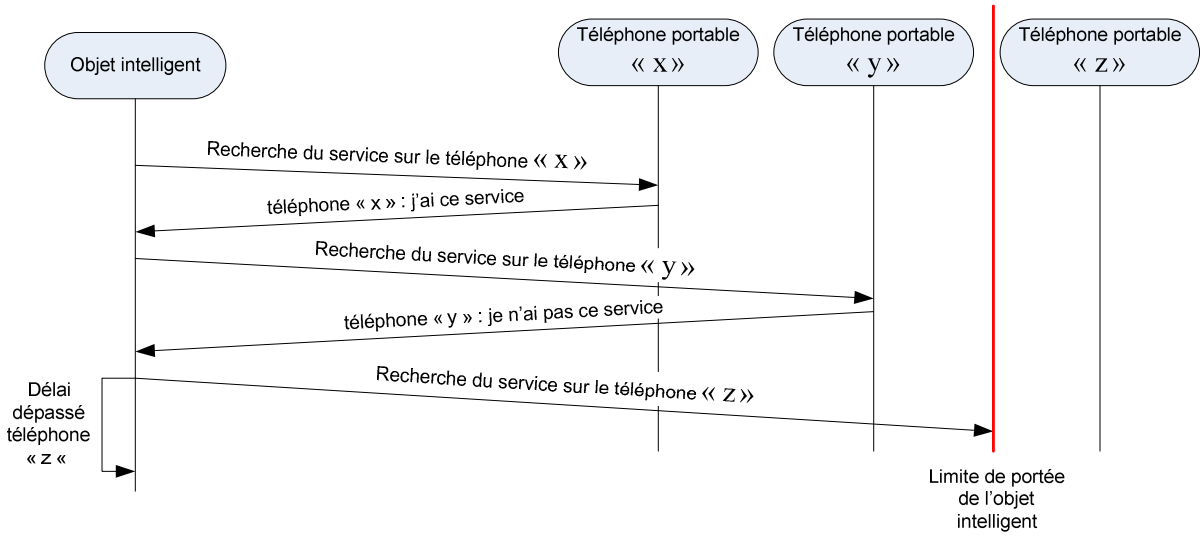


Figure 22 : Recherche du service spécifique

D'après ce résultat, on peut en tirer les conclusions suivantes :

Téléphone portable	A portée de l'objet intelligent ?	Met à disposition le service « Remote Control » ?
« X »	OUI	OUI
« Y »	OUI	NON
« Z »	NON	Pas d'information

Figure 23 : Résultat de la recherche de service

La connexion peut s'établir avec le téléphone « x » uniquement.

A.4. RSSI

RSSI (Receiver Signal Strength Indicator) est le terme qui définit la puissance à laquelle l'émetteur reçoit le signal provenant d'un autre appareil connecté.

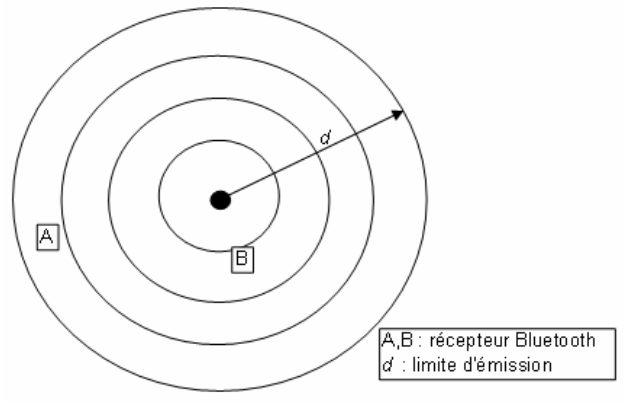


Figure 24 : RSSI

Deux dispositifs Bluetooth **A** et **B** qui souhaitent participer à une liaison puissance-commandé peuvent mesurer la puissance à laquelle le signal est reçu. Ils déterminent ensuite si l'émetteur ● doit augmenter ou diminuer son niveau de puissance de sortie. Un indicateur de puissance de signal de réception (RSSI) rend ceci possible.

La mesure du RSSI en A est plus faible qu'en B dans ma Figure 24 : RSSIFigure 24 ci-dessus.

La modification de la puissance d'émission de ● a pour but de recevoir en **A** ou **B** un signal dans une gamme de puissance déterminée est de faciliter la lecture du signal reçu. Les instructions qui permettent de changer la puissance de l'émetteur ● sont passées à la couche du gestionnaire de liaison.

A.5. L'appariement

Pour pouvoir connecter deux appareils Bluetooth, il faut que l'un d'eux scanne autour de lui et liste tous les appareils Bluetooth activés.

Deux étapes principales sont à suivre : l'un des dispositifs doit découvrir l'autre et ils doivent échanger des mots de passe de sécurité afin de devenir appariés.

La nécessité d'effectuer l'appariement avant d'établir une connexion dépend du profil utilisé (détails annexe A.3).

L'appariement dans ce projet

Le téléphone portable met à disposition une fonction qui permet d'effectuer la recherche d'appareil Bluetooth et la demande d'échange de mot de passe. En revanche l'appariement doit être implémenté dans le Casira.

Le processus se déroule comme ceci :

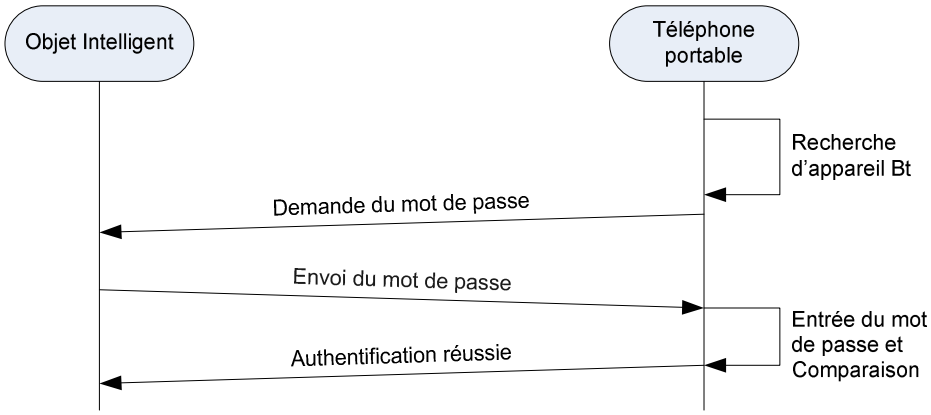


Figure 25 : L'appariement

Lorsque l'authentification est réussie, l'adresse et le nom du téléphone sont sauvegardés en mémoire pour permettre une connexion dans le futur. La programmation qui permet de gérer ce procédé en C est disponible en annexe J.2.

A.6. Maître / Esclave, Client / Serveur

Chaque appareil Bluetooth qui utilise une connexion est définissable d'après deux de ces critères. La notion de client / serveur est valable d'un point de vue Software, tandis que la notion de maître / esclave l'est côté Hardware.

Hardware

Le maître est celui qui contrôle une liaison Bluetooth directe ou un piconet. Son horloge interne sert de référence, elle est appelée horloge maîtresse. Les esclaves doivent se synchroniser à l'horloge du maître. Cette synchronisation permet aux intervalles d'émission et de réception de l'esclave de se mettre en phase avec ceux du maître. Sans cette synchronisation, les deux appareils risquent d'émettre en même temps.

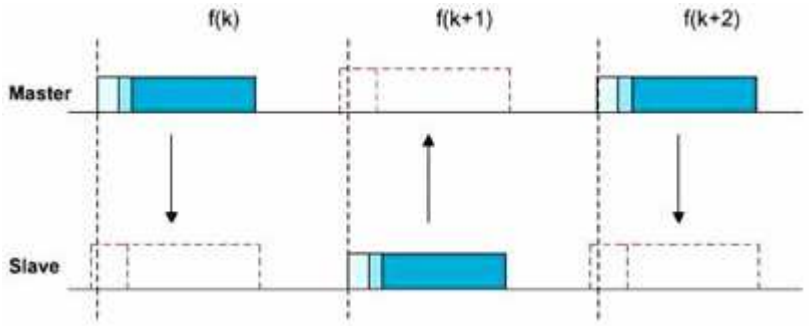


Figure 26 : maître / esclave

Software

Un serveur Bluetooth met à disposition des profils. Un client peut se connecter à ses profils. Les profils ont pour but d'assurer une interopérabilité entre tous les appareils Bluetooth. Une explication supplémentaire concernant les profils est disponible en annexe A.3.

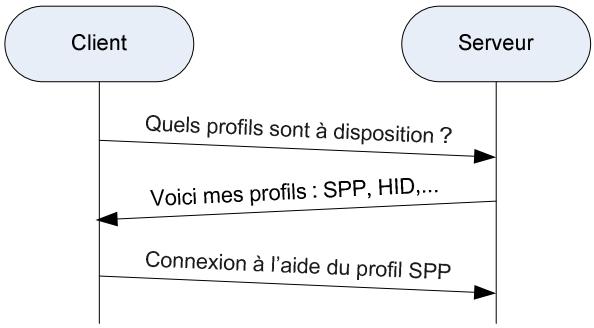


Figure 27 : client / serveur

A.7. Type d'appareil

La norme Bluetooth définit un champ (Class of Device) capable de classer les appareils Bluetooth par catégories. Elle prévoit de définir toutes sortes d'appareils domestiques ou industriels qui sont orientés multimédia, informatique ou en rapport à la téléphonie mobile. Il est par exemple possible de définir précisément un écran d'ordinateur, un kit main libre ou encore un autoradio. En revanche, la spécification est incapable de définir précisément des appareils tels qu'une porte de garage, un chauffage, une machine à café…

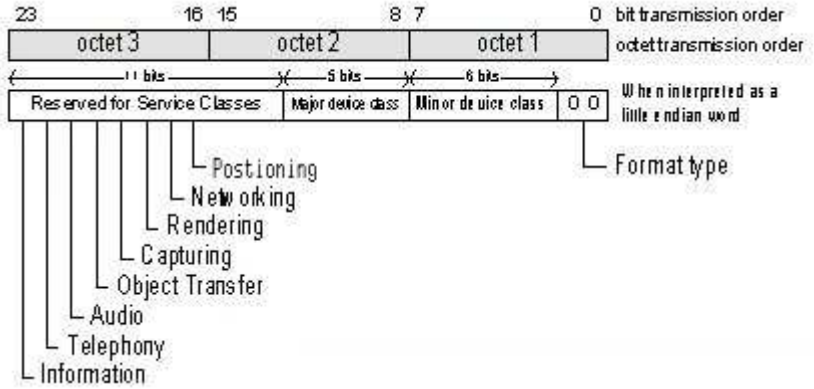


Figure 28 : Classe de l'appareil

Chaque périphérique Bluetooth est défini selon trois niveaux de détails. La procédure complète qui détermine les niveaux est disponible ici : https://www.bluetooth.org/assigned-numbers/baseband.php

B. Environnement de développement

Le matériel mis à ma disposition est composé :

- d'un téléphone portable
- d'un système de développement Casira
- d'un circuit Bluetooth CSR BlueCore 2
- d'une suite de logiciels.

Le téléphone portable est programmé en java (raisons en 2.1), tandis que le Casira est programmé en C ANSI.

B.1. Système Casira

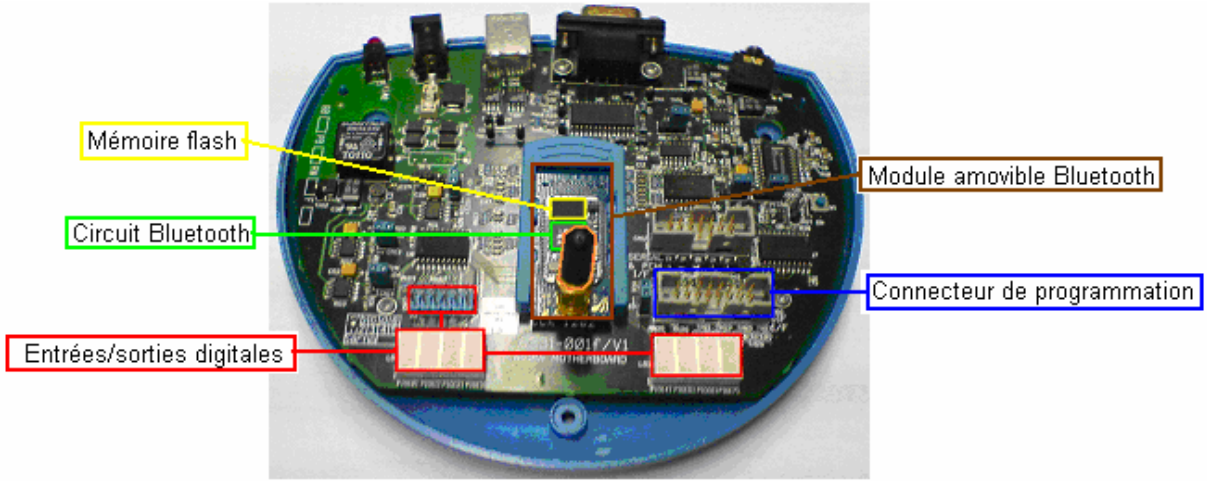


Figure 29 : Casira en détails

Le système met à disposition 8 entrées/sorties digitales. Huit leds peuvent être contrôlées directement. Un connecteur permet d'accéder aux entrées/sorties. Sa connectique est définie comme ceci :

Pin2 : PIO[0]	Pin4 : PIO[2]	Pin6 : PIO[4]	Pin8 : PIO[6]	Pin10 :	Pin12 : GND
Pin1 : GND	Pin3 : PIO[1]	Pin5 : PIO[3]	Pin7 : PIO[5]	Pin9 : PIO[7]	Pin11 :

Le système dispose de plus d'un port RS232, d'un port USB ainsi que d'un connecteur audio.

Le système Casira permet la programmation du module amovible Bluetooth (annexe B.1.1) en C ANSI à l'aide du programme BlueLab décrit en annexe B.5.

B.1.1. Module amovible Bluetooth

Figure 30 : Module amovible Bluetooth

Le circuit CSR BlueCore 2 est un processeur ARM7 radio et de bande de base pour le système Bluetooth 2.4Ghz.

La documentation complète est disponible sur le site du constructeur à l'adresse http://www.csr.com/products/bc2range.htm.

Grâce aux librairies mises à disposition dans l'IDE Bluelab, j'ai la possibilité d'accéder à énormément de paramètres bas niveau du CSR BlueCore 2.

Les libraires qui m'intéressent particulièrement sont :
- Spp : permet d'établir une connexion Bluetooth de type SPP
- Vm : permet notamment d'accéder aux paramètres de puissance d'émission
- Connection : permet de récupérer des informations sur la qualité du signal reçu
- Ps : permet d'accéder en lecture/écriture aux registres utilisateur
- Sink : permet de transmettre des messages à la couche Bluetooth
- Source : permet d'obtenir des messages de la couche Bluetooth

B.1.2. Puissance d'émission

La librairie Vm permet d'accéder aux différents paramètres de puissances d'émission :
- Valeur par défaut
- Valeur maximale
- Table des puissances consommation normale
- Table des puissances en basse consommation

Les valeurs par défaut et maximale sont accessibles grâce aux méthodes suivantes :

```
void VmTransmitPowerSetMaximum(int);
int VmTransmitPowerGetMaximum(void);

void VmTransmitPowerSetDefault(int);
int VmTransmitPowerGetDefault(void);
```

Code 1 : VmTransmitPower

L'interface graphique (PSTool annexe B.3, Figure 34) permet de visualiser les deux tables de puissance. Elles sont déclarées comme un tableau de structures légèrement différentes. Chaque ligne du tableau est définie comme ceci :

```
struct PSKEY_LC_POWER_TABLE
{
    uint8 external_pa   /* Paramètre de puissance externe */
    uint8 internal_pa   /* Paramètre de puissance interne */
    uint8 zero
    int8  tx     /* valeur de la puissance d'émission en complément à 2 */
};
```

Code 2 : structure PSKEY_LC_POWER_TABLE

La structure PSKEY_LC_LOWPOWER_TABLE est définie comme la structure précédente à la différence du byte fine_param :

```
struct PSKEY_LC_LOWPOWER_TABLE
{
    uint8 external_pa  // Paramètre de puissance externe */
    uint8 internal_pa  // Paramètre de puissance interne */
    uint8 fine_param   // Paramètre concernant des modification très fines
                       // de la puissance ainsi que du filtre passe bande
                       // qui sélectionne le canal de transmission.
    int8  tx           // valeur de la puissance en complément à 2
};
```

Code 3 : structure PSKEY_LC_LOWPOWER_TABLE

B.1.3. Antennes

Différents types d'antennes peuvent être utilisées sur le module amovible Bluetooth grâce à un connecteur standard.

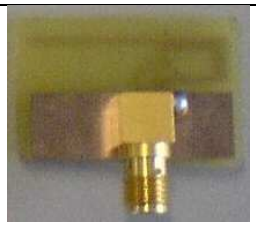
Antenne N° 1	Antenne N° 2	Antenne N° 3	Antenne N° 4
			

Figure 31 : Différentes antennes

B.2. Téléphone portable

Le téléphone portable est l'interface qui me permet d'agir sur le Casira ou sur l'objet intelligent. Ceci est fait grâce à une application Java.

Pour être capable d'exécuter une application Java Bluetooth, le téléphone portable doit supporter les éléments suivants :

- Technologie Bluetooth
- J2ME, configuration CLDC (Annexe D.3)
- JSR82 (Annexe D.1)

La liste des téléphones portables actuels qui disposent de ces technologies est accessible à l'annexe C.

Le développement de l'application Java est réalisé à l'aide du Siemens SK65. L'application est ensuite testée sur différents types de téléphones portables.

Figure 32 : Siemens SK65

La connexion avec le système de développement Casira via le profile SPP nécessite l'appariement. Le SK65 permet de réaliser cette tâche comme ceci :

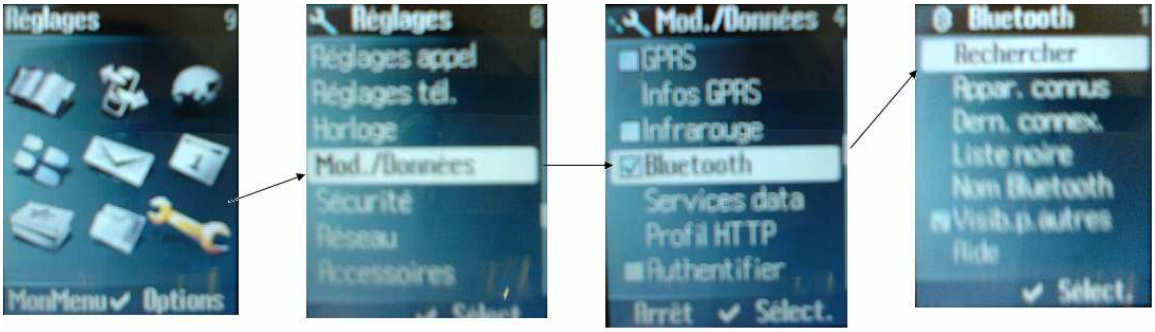


Figure 33 : Siemens SK65: recherche d'un appareil Bluetooth

Lorsque l'appariement est effectué, il est possible de lancer l'application Java et d'établir la connexion entre le téléphone portable et mon système de développement.

B.3. PSTool

Description

Le système de développement met à disposition un logiciel qui permet d'accéder en lecture/écriture via une interface graphique aux différents registres du circuit CSR BlueCore 2. Il permet un affichage du nom des registres ainsi qu'à une brève description de chacun d'entre eux.

Utilisation

Ci-dessous sont détaillés les registres du module Casira qui me seront utiles lors du développement de mon application.

Adresse mémoire	*Dénomination*	*Type*	*Commentaire*
0017	Maximum Transmit power	R/W	Valeur par defaut : 20 *
001E	Radio power table	R/W	
0021	Default transmit power	R/W	Valeur par defaut : 4 *
0022	Radio power table low power mode	R/W	
01F9	Host interface	VM access to the Uart	Permet le mode de débuggeur
0294	User Config XX	R/W	Etat de la connexion (apparié, connecté)

* Il est théoriquement possible de changer cette valeur. En réalité la modification de ce paramètre n'a aucune influence sur le système. Par conséquence l'affichage de cette valeur n'est pas utilisable.

Les paramètres Radio power table et Radio tx power table in low power mode permettent de visualiser et de modifier les tableaux des puissances que va utiliser le processeur pour alimenter l'antenne Bluetooth. Ces deux tables sont directement liées avec la portée qu'aura le nœud Bluetooth. Lors de l'assignement d'une valeur de puissance dans le module Casira, le processeur prendra la valeur la plus proche inscrite dans les tableaux suivants. (Champ dBm)

Internal PA	External PA	dBm
15	0	-20
20	0	-16
25	0	-12
32	0	-8
40	0	-4
50	0	0
57	0	4
63	0	6

Internal PA	External PA	dBm
4	0	-50
6	0	-48
9	0	-46
12	0	-44
15	0	-42
21	0	-40
25	0	-38
31	0	-36
37	0	-34
43	0	-32
49	0	-30

Figure 34 : Power Table

La table de gauche est en mode normal, celle de droite en mode basse puissance.

Un module d'amplification externe peut être ajouté avant d'attaquer l'antenne avec le signal radio. Le module Casira ne dispose pas d'un tel module.

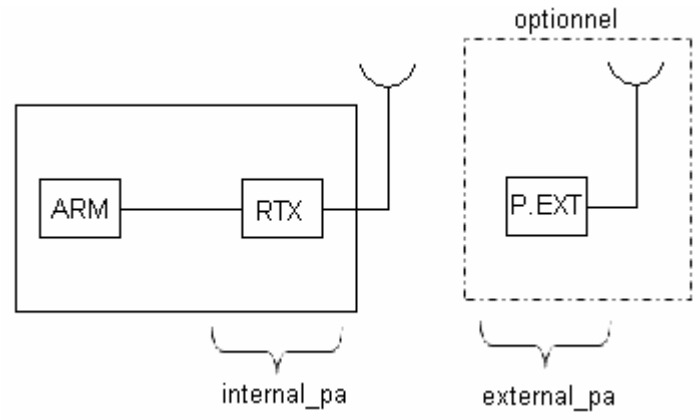


Figure 35 : Amplification du signal radio

B.4. BlueFlash

Cet utilitaire donne la possibilité d'effacer complètement le processeur et de recharger un firmware d'usine. Il sera utilisé lors d'une difficulté de la transmission du PC vers le Casira. Lors d'un problème de ce type il n'est plus possible de programmer le processeur en utilisant le logiciel Bluelab. En rechargeant le firmware, la programmation est à nouveau accessible.

B.5. BlueLab v3.5.2

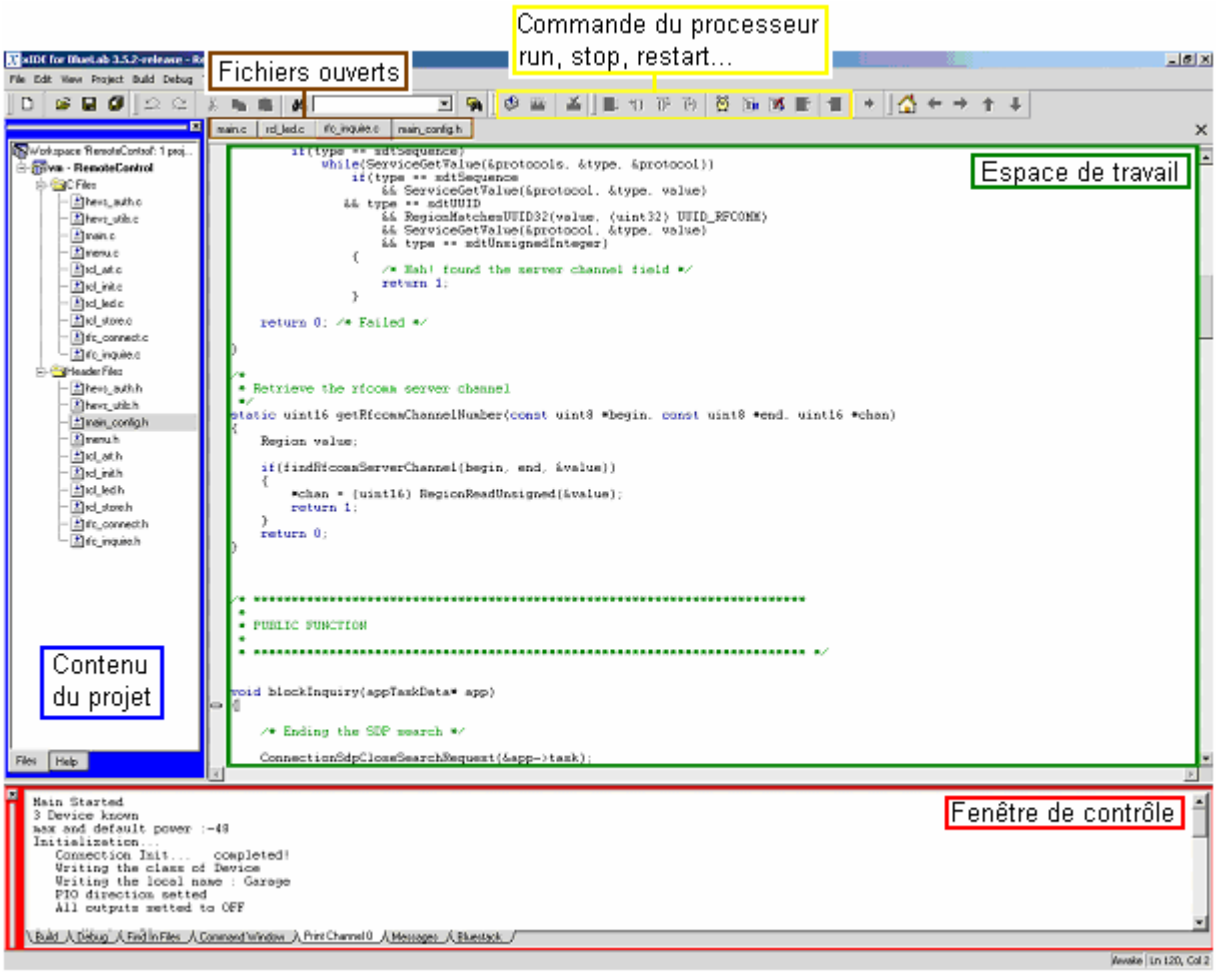


Figure 36 : BlueLab v3.5.2

Ce programme est utilisé pour le développement d'application en C ANSI. Dans un deuxième temps, il permet la programmation du processeur. Une aide détaillée au format html est mise à disposition.

B.6. Carte de commandes (maquette)

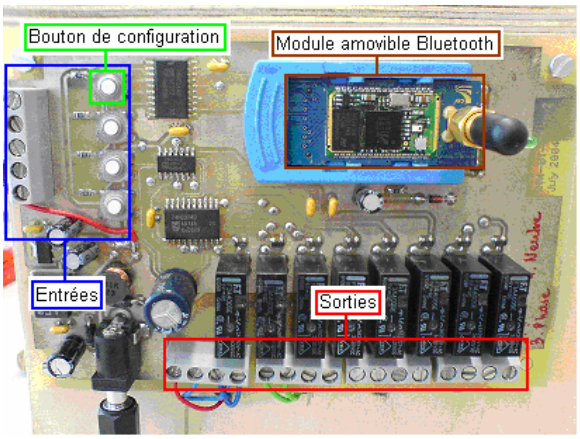


Figure 37 : Carte de commandes

La commande des objets de la maquette est réalisée grâce à une carte électronique. Elle dispose de 4 entrées numériques et 8 relais (220VAC/0.3A) de sorties. Deux cartes de commandes sont intégrées à la maquette. Elles sont connectées aux objets et leur donne une intelligence.

Comme vu au point **Erreur ! Source du renvoi introuvable.**, l'objet intelligent dispose d'un bouton de configuration. Celui-ci permet de choisir entre le mode « appariable » et « recherche ».

Garage

Un moteur permet d'ouvrir et fermer la porte. Celui-ci est connecté à deux sorties de la carte commande. Une troisième sortie est reliée à la lampe qui se situe devant le garage.

Un capteur à effet hall est branché à une entrée pour déterminer si le garage est libre ou non.

Télévision

La télévision est un PDA déguisé connecté par Bluetooth à la carte de commandes. Une application a été développée au sein de l'unité infotronics pour communiquer avec la carte de commande et afficher des images.

C. Liste des téléphones qui supportent JSR82

La liste ci-dessous est tirée du site :
http://www.club-java.com/TastePhone/J2ME/MIDP_Benchmark.jsp
Ce site très complet sur les possibilités et caractéristiques des téléphones portables et mis à jour régulièrement. Elle est datée ici du 22 novembre 2006.

Marque	Modèle	Arrière plan	Marque	Modèle	Arrière plan	Marque	Modèle	Arrière plan
LG	LX550	Oui	Nokia	6682	Oui	Siemens	M75	Oui
Motorola	A1000	Oui	Nokia	7610	Oui	Siemens	S65	Non
Motorola	A-5B	Oui	Nokia	7710	Oui	Siemens	S6V	Non
Motorola	C975	Non	Nokia	9500	Oui	Siemens	SK65	Non
Motorola	V360	Oui	Nokia	N70-1	Oui	Siemens	SX1	Oui
Nokia	3230	Oui	Nokia	N90-1	Oui	Siemens	SXG75	Oui
Nokia	3250	Oui	Panasonic	X701	Oui	SonyEricsson	K600i	Oui
Nokia	5500d	Oui	Samsung	SGH-D347	Oui	SonyEricsson	K750i	Oui
Nokia	6021	Non	Samsung	SGH-D357	Oui	SonyEricsson	K790i	Oui
Nokia	6102i	Non	Samsung	SGH-D407	Oui	SonyEricsson	K800i	Oui
Nokia	6230	Non	Samsung	SGH-D800	Oui	SonyEricsson	P900	Oui
Nokia	6230i	Non	Samsung	SGH-D807	Oui	SonyEricsson	P910	Oui
Nokia	6255	Non	Samsung	SGH-D900	Oui	SonyEricsson	P910a	Oui
Nokia	6260	Oui	Samsung	SGH-E900	Oui	SonyEricsson	P910i	Oui
Nokia	6280	Non	Samsung	SGH-X507	Oui	SonyEricsson	P990i	Non
Nokia	6600	Oui	Samsung	SGH-X810	Oui	SonyEricsson	W800i	Oui
Nokia	6620	Oui	Sendo	X	Oui	SonyEricsson	W810i	Oui
Nokia	6630	Oui	Siemens	C65	Oui	SonyEricsson	W850i	Oui
Nokia	6670	Oui	Siemens	CX70	Oui	SonyEricsson	Z520i	Oui
Nokia	6680	Oui	Siemens	CX75	Non	SonyEricsson	Z530i	Oui
Nokia	6681	Oui	Siemens	EF81	Oui			

Lorsque une application Java est exécutée, elle utilise la totalité de l'écran du téléphone. La colonne « arrière plan » indique s'il est possible de la minimiser. Ceci rend accessible les fonctionnalités standard du téléphone tout en gardant l'application Java lancée.

D. JAVA

D.1. JSR82

La JSR82 est un ensemble d'API standardisé qui permet le développement d'applications dans un environnement Bluetooth. Cette spécification inclut un support de base pour utiliser les profils tel que RFCOMM, OBEX, SDP… La possibilité est offerte de déclarer et mettre à dispositions de nouveaux profils.

D.2. Midlet et Canvas

La classe Midlet et la classe Canvas sont deux classes qui définissent de quelle manière l'écran du téléphone sera utilisé par une application java.

Classe Midlet (API haut niveau)

La programmation d'une application graphique java à l'aide de la classe Midlet est relativement simple. Toute une série d'élément sont prédéfinis par J2ME et peuvent être intégré à l'écran. J2ME est une version allégée des classes Java pour une implémentation en systèmes embarqués.

Exemple d'une Midlet comprenant différent type d'éléments :

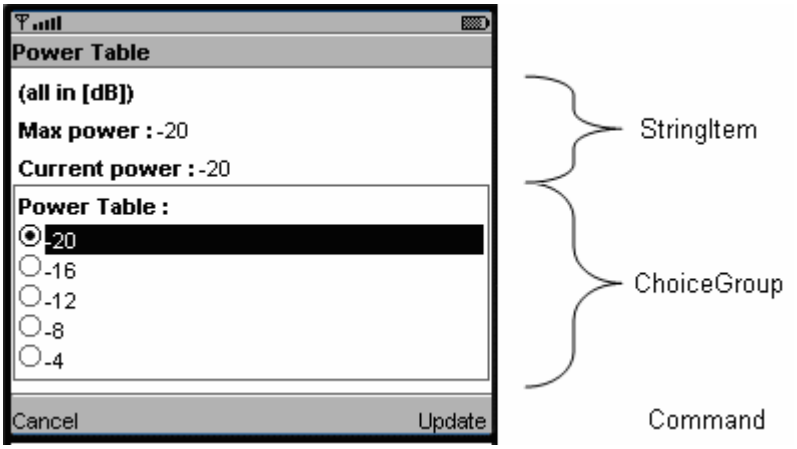


Figure 38 : Exemple de midlet

Les éléments prédéfinis en java (p.ex `StringItem`, `ChoiceGroup)` peuvent être affichés de manière légèrement différente d'un téléphone à l'autre.

Classe Canvas (API bas niveau)

Un objet de la classe Canvas représente un carré blanc sur l'écran dans lequel l'application peut afficher des images et gérer des événements produits par l'utilisateur.

Le grand avantage de cette classe est qu'elle permet de créer un type d'application graphique complètement indépendant du téléphone qui l'exécute.

Il n'est pas possible dans un Canvas d'intégrer des éléments de type `Choicegroup` ou des boutons prédéfinis en java comme dans une midlet. Par exemple, pour créer un bouton, l'application a besoin de deux images (bouton appuyé, relâché). Toutes les images nécessaires à l'affichage correct de l'application doivent être enregistrées dans l'application, cela fait que celle-ci est relativement lourde.

Une télécommande de télévision où chaîne stéréo pourrait ressembler à ceci :

Figure 39 : Exemple de canvas

Conclusion

Pour disposer d'une application java complètement générique du point de vue graphique, l'utilisation des Canvas est obligatoire. Par contre, le fait qu'avec les Canvas, il faille tout gérer prend plus de temps à programmer. L'application est certes très ''jolie'' mais pour démontrer la faisabilité du projet, cela n'est pas nécessaire.

Les téléphones qui supportent les applications utilisant la classe Canvas supportent également la classe Midlet, mais l'inverse n'est pas forcément vrai. Par conséquent, une plus grande quantité de téléphones peut utiliser mon application si je la développe à l'aide des Midlets.

Je vais utiliser les midlets dans mon travail de diplôme pour les raisons citées ci-dessus et sur conseils d'un assistant qui connaît les deux aspects des classes graphiques.

D.3. CDC et CLDC

CDC et CLDC sont deux configurations de l'API java J2ME.
CDC est à destination d'appareil possédant des processeurs 32 bits, 2 MO de RAM et 2,5 MO de ROM pour l'environnement. Ces appareils disposent aussi d'une connexion au réseau. Ce sont des appareils tels que les PDA haut de gammes, décodeur télévision, ...
Pour plus de détails, consulter : http://java.sun.com/products/cdc/

CLDC est à destination des appareils à puissance modeste tel que les téléphones portables.
Pour plus de détails, consulter : http://java.sun.com/products/cldc/

La configuration utilisée lors du projet est par conséquent CLDC.

D.4. MIDP 2.0

Abréviation de ***M**obile **I**nformation **D**evice **P**rofile*, MIDP est un ensemble d'API J2ME qui définit la façon dont les applications de logiciel se connectent à l'interface des téléphones cellulaires. Les applications conformes à cette norme s'appellent MIDlets. Cette norme peut aussi être utilisée pour développer des applications sur des PDA de type Palm. Les principales sociétés ayant travaillé au MIDP sont les grands constructeurs de téléphonie mobile ainsi que Palm Computing.

Les possibilités concernant l'IHM (Interface Homme Machine) de MIDP sont très réduites pour permettre une exécution sur un maximum de machines. Avec un clavier limité en

nombres de touches et dépourvu de système de pointage, la saisie de données sur de tels appareils est particulièrement limitée.

L'API du MIDP se compose des API du CDLC et de trois packages :

- javax.microedition.midlet : cycle de vie de l'application
- javax.microedition.lcdui : interface homme machine
- javax.microedition.rms : persistance des données

Lors de mon travail, je vais utiliser uniquement les packages midlet et lcdui. Le package rms ne m'est pas utile car il n'est pas nécessaire dans mon application de sauvegarder des données dans le téléphone portable. La seule donnée qui doit rester enregistrée dans le téléphone portable est l'adresse de l'objet intelligent. Cette adresse est enregistrée lors de l'appariement des deux appareils. Ceci est fait par le téléphone portable et non par mon programme.

Midlet et Lcdui gèrent l'affichage sur le téléphone portable ainsi que l'utilisation de commandes en passant par le clavier.

E. Protocole de communication

Une application développée pour le téléphone portable me permet d'interagir avec le système embarqué et d'en modifier les paramètres de puissance d'émission. La communication entre le téléphone portable et le Casira se fait selon le schéma suivant :

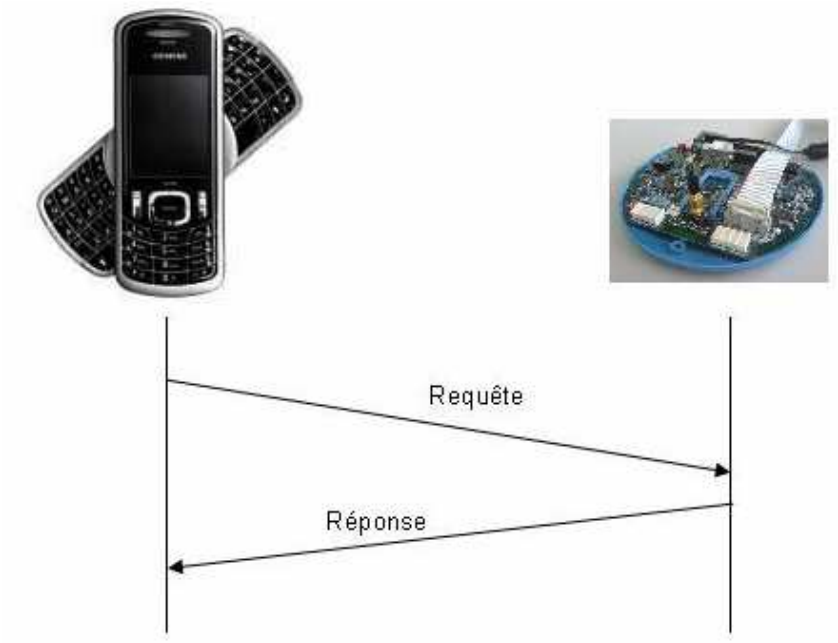


Figure 40 : Requête - Réponse

La requête est un String composé des éléments suivants :

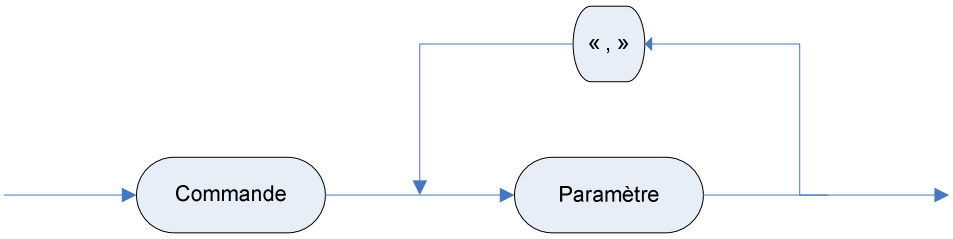
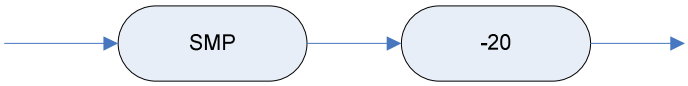


Figure 41 : Structure d'une requête

L'élément Commande contient trois caractères. L'élément Paramètre est de longueur quelconque.

Par exemple la requête suivante permet d'assigner la nouvelle valeur de puissance d'émission désirée dans le Casira.



La réponse à une requête est un élément de type `String` composé des éléments suivants :

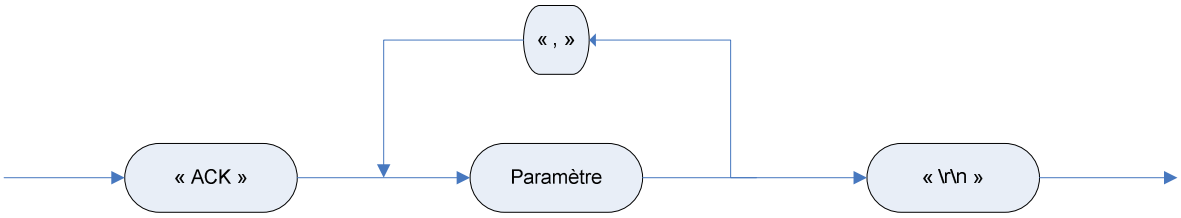
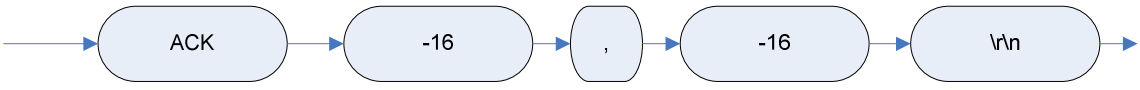


Figure 42 : Structure d'une réponse

Par exemple la réponse suivante contient la valeur de la puissance maximale et courante lue dans le BlueCore 2. Le string lu par le programme Java est « ACK-16,-16 ». Il est très facile de traiter ce string pour l'affichage sur le Siemens.



La réponse est nécessaire pour valider la bonne exécution de la commande. Il se peut que la communication soit perdue ou que la requête soit mal comprise par le BlueCore 2.

F. Mesure de proximité, application Java

Le but est d'offrir une interface simple d'accès aux registres qui concernent la puissance du Casira. L'application dispose également de plusieurs fonctionnalités permettant de réaliser des mesures de portée avec le Casira.

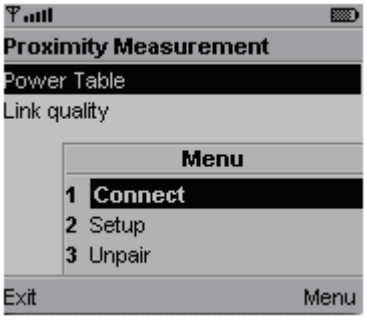


Figure 43: Formulaire de démarrage

Description des éléments :
1. Connect : établit la connexion avec le module Casira
2. Setup : affiche le nom et l'adresse du système embarqué
3. Power Table : donne accès au registre du module en lecture / écriture
4. Link quality : affiche la qualité de la connexion entre les deux appareils

Formulaire « Power Table »

Ce formulaire permet d'afficher le contenu des paramètres de puissance du BlueCore 2 tel qu'il apparaît dans ses registres.

Lors de l'affichage du formulaire java PowerTable, deux requêtes sont envoyées. La première reçoit dans la réponse les valeurs courantes de la puissance et le maximum admissible. La seconde reçoit toutes les valeurs de la table contenant les paramètres de puissance.

La liste des valeurs est créée dynamiquement en fonction de la seconde réponse.

Il est possible de changer la puissance maximale et courante en choisissant un champ de la table et en sélectionnant update.

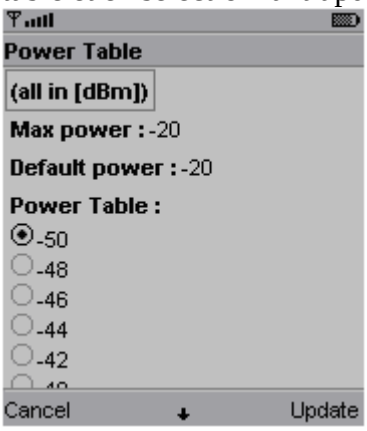


Figure 44 : Formulaire Power Table

Affiche les valeurs utilisées par le module Casira. Toutes ces valeurs sont le résultat d'une lecture des registres du module Casira.

La commande update permet d'accéder en écriture aux registres du module Casira.

Formulaire « Link quality »

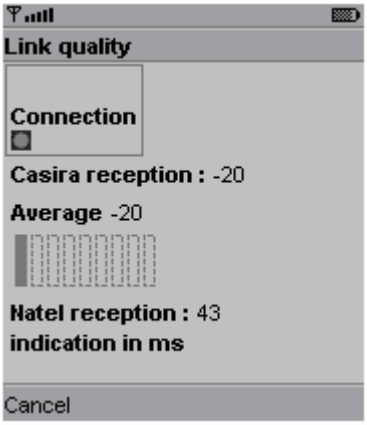
Ce formulaire a pour objectif de montrer clairement l'état de la connexion ainsi que la puissance avec laquelle le Casira voit le téléphone portable.

Affichage de l'état de la connexion

Il y a trois états possibles pour la connexion. Soit elle fonctionne normalement, soit la connexion est active, mais la transmission n'est pas possible. Ce deuxième cas apparaît lorsqu'on vient de quitter la zone d'émission du Casira. Bluetooth garde la connexion active quelques secondes avant de déconnecter. Le troisième état est déconnecté.

Affichage du RSSI

Cette valeur est une indication de la manière dont le Casira perçoit le téléphone portable. Elle n'est pas essentielle pour la mesure de la portée du Casira mais, est une indication sur la qualité d'émission du téléphone portable.



Affiche l'état de la connexion :
- : Connexion active, la communication fonctionne
- : Connexion active, la communication ne fonctionne pas
- ×: Déconnecté

Une fois que la communication est perdue, elle est encore active quelques secondes. Il est possible de la retrouver en se rapprochant de l'antenne.

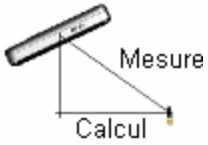
Figure 45 : Formulaire Link quality

Les champs Casira réception et Natel réception sont uniquement une indication sur la qualité de la connexion entre le téléphone portable et le Casira. Ces deux champs ne sont pas utilisés lors des mesures de portées.

G. Résultat complet des mesures de portée

Mesure effectuées à Tourtmagne le 21.09.2006 de 9h30 à 13h30.

						Distance limite[m]					
						Mesure					
ID	Niveau bat	Type	N° d'antenne	Position de l'antenne	Puissance du Casira [dBm]	1	2	3	4	Moyenne	remarques
1	90%	SK65	2	vertical	6	126	125	128		126.3	
2		SK65	2	vertical	-50	0.45	0.6				1:*, 2:**
3		SK65	2	vertical	-46	0.6	1.45				1:*, 2:**
4		SK65	2	vertical	-40	2.5	2.45	2.5		2.5	
5		SK65	2	vertical	-36	5.25	5	5.1		5.1	
6		SK65	2	vertical	-32	8.2	8.3	8.3		8.3	
7		SK65	2	vertical	-34	8	7.7	8		7.9	
8		SK65	2	vertical	-30	9.15	8.8	9.3		9.1	
9		SK65	2	vertical	-20	48	40.5	48.5		45.7	-
10	50%	K750i	2	vertical	-50	0.27	0.27				1:*, 2:**
11		K750i	2	vertical	-46	0.5	0.3				1:*, 2:**
12		K750i	2	vertical	-40	0.36	0.8				1:***, 2:**
13	40%	K750i	2	vertical	-36	0.7	0.9				1,2 :**
14		K750i	2	vertical	-32	1	2.5	2	2.5	2.0	
15		K750i	2	vertical	-34	0.8	0.8	0.8		0.8	
16		K750i	2	vertical	-30	2.5	2.3	2	2.5	2.3	
17		K750i	2	vertical	-20	30	36.5	30		32.2	
18		K750i	2	vertical	6	103	110	118.5		110.3	
19	25%	K750i	2	vertical	-30	3.3	4.7	3.2		3.7	
20		K750i	1	vertical	-30	2.5	1.8	1.7		2.0	

* :  ** :  *** : 

Valeur calculée d'après la mesure.

ID	Niveau bat	Type	N° d'antenne	Position de l'antenne	Puissance du Casira [dBm]	Distance limite [m] Mesure 1	2	3	4	Moyenne	remarques
21		K750i	4	horizontal	-30	1.7	1.75	1.6		1.7	
22		K750i	3	vertical	-30	4.5	4.6	5		4.7	
23		K750i	4	horizontal	-30	1.1	1	1.2		1.1	
24		K750i	4	horizontal	6	36	37	36		36.3	
25		K750i	4	horizontal	-30	1.3	1.3	1.2		1.3	
26		K750i	3	vertical	6	129					
27		K750i	3	vertical	-30	4.3	3.3	3.5		3.7	
28	90%	SK65	3	vertical	6	129	126			127.5	
29		SK65	3	vertical	-30	9.3	8.3	8.5	7.5	8.4	
30		SK65	4	horizontal	-30	1.7	1.9	1.5		1.7	
31		SK65	1	vertical	-30	4.8	4.9	4.7		4.8	
32		SK65	2	vertical	-30	8.7	8.7	8.8		8.7	

ID *1*, *28* : Deux types de mesures identiques à deux moments très différents ont produits des résultats avec un écart d'environ 1%.

ID *16*, *18*, *19* : *18* est conforme à mes attentes, Le résultat de *19* est environ 150% du résultat *16*.
ID *22*, *27* : Deux mesures type de mesure identique. Le résultat de *27* est 20% inférieur à *22*.
L'écart important qui existe entre *16* et *19* ou *22* et *27* dépend de paramètres que je n'ai pas réussi à déterminer. Malgré que j'essaie de produire les mesures en ne changeant que le minimum de paramètres, il est possible que ceux-ci soient différents :

- Position de la main sur le téléphone
- Inclinaison du téléphone

H. Comportement de différents téléphones avec le système

H.1. Siemens SK65

Perte de connexion à cause de la portée :

La déconnexion est détectée par le téléphone portable et l'objet intelligent. L'objet recommence le processus de recherche de téléphones appariés. Le téléphone est trouvé s'il est à nouveau à portée. Si tel est le cas, la connexion est ouverte et les commandes peuvent être envoyées.

Fermeture du programme :

La déconnexion est détectée par l'objet intelligent. Il recommence son processus de recherche de téléphones appariés.

Cas particulier à ce type de téléphone (connexion délai dépassé):

Un message de type ''connection timeout'' est reçu par l'objet intelligent dans le cas où l'utilisateur ne répond pas à la demande de connexion du téléphone portable.

Après deux connexions timeout, la demande de connexion ne se fait plus. Il faut redémarrer le téléphone pour pouvoir accepter une connexion.

Cas particulier à ce type de téléphone (connexion rejetée):
Un message de type ''connection rejected'' est reçu par l'objet intelligent dans le cas ou l'utilisateur refuse la permission à l'objet de se connecter au téléphone portable.

Après deux connexions rejetées, l'objet intelligent effectue sa recherche d'appareil à portée.

Lorsque le téléphone est retrouvé, l'objet intelligent essaie d'établir la connexion, mais le téléphone n'envoie pas les messages permettant de terminer le processus.

H.2. Sony Ericsson K750i

Perte de connexion à cause de la portée :

La déconnexion est détectée par le téléphone portable et l'objet intelligent. L'objet recommence le processus de recherche de téléphones appariés. Le téléphone est trouvé s'il est à nouveau à portée. Si c'est le cas, la connexion est ouverte et les commandes peuvent être envoyées.

Fermeture du programme :
La déconnexion est détectée par l'objet intelligent. Il recommence son processus de recherche de téléphones appariés.

Application minimisée:

Ce type de téléphone permet de minimiser l'application et de continuer à utiliser le téléphone normalement. L'application est mise au premier plan lorsque la connexion est établie ou lorsque la connexion est perdue.

I. Code Java

I.1. Url Bluetooth

L'url est une adresse qui permet d'établir une connexion entre deux appareils Bluetooth. Elle contient des paramètres différents si le téléphone portable est programmé en client ou en serveur, à l'exception du service utilisé pour établir la connexion.

La spécification Bluetooth définit le champ ‘’type de connexion‘’ possible :

- btspp : Connexion à l'aide du profile SPP
- btgoep : Connexion à l'aide du profile OBEX

Ce champ est lié aux couches Bluetooth définies en A.2.

I.1.1. Client

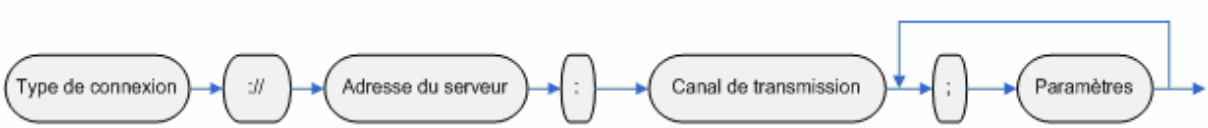


Figure 46 : Url Bluetooth client

L'adresse du serveur correspond à l'adresse Bluetooth sans les séparateurs conventionnels.
Exemple d'adresse Bluetooth :

- représentation standard : 00:01:e3:44:ff:9a
- format pour le champ de l'url : 0001e344ff9a

Bluetooth permet d'établir plusieurs connexions à un même serveur Bluetooth, il faut donc choisir un canal qui permet d'aiguiller où les données sont traitées.

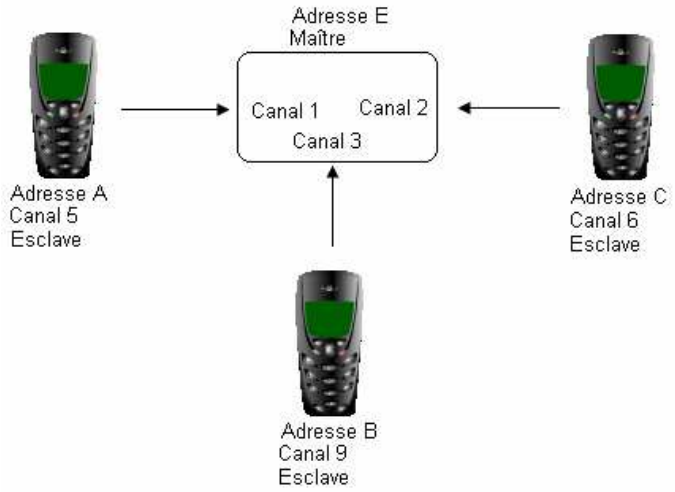


Figure 47 : Adresse Bt, canal

Le dernier champ peut contenir différents paramètres de type booléen :

- `encrypt` : Spécifie que la connexion doit être encryptée
- `authenticate` : détermine si l'échange de mot de passe est nécessaire pour établir la connexion.
- `master` : Indique quel appareil est le master d'un réseau d'éléments Bluetooth.

Exemple d'une url client :

```
btspp://00025B00A5:1;encrypt=false;authenticate=false;master=false
```

Code 4 : Url Bluetooth client

I.1.2. Serveur

L'url serveur permet de savoir de quelle manière le service mis à disposition par un serveur peut être atteint.

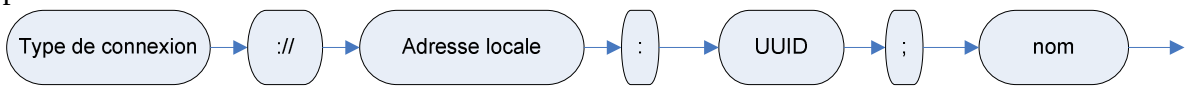


Figure 48 : Url Bluetooth serveur

L'UUID est un identifiant unique pour chaque profil déclaré.

Exemple d'une url serveur :

```
btspp://localhost:UUID;name=Remote Control
```

Code 5 : Url Bluetooth serveur

I.2. Connexion client

Avant de pouvoir établir une connexion client entre deux appareils, il est nécessaire que celui qui demande la connexion connaisse l'adresse du second appareil. Celle-ci est recherchée grâce au profil Bluetooth SDP. L'adresse est sauvée et sera utilisée plus tard pour demander la connexion.

La connexion est demandée grâce à une chaîne de caractère appelée « url ». (Annexe I.1.1) Les différents codes java ci-dessous montrent la procédure nécessaire à ouvrir une connexion. Ils correspondent à l'organigramme de la Figure 4.

```
/** Récupère l'adresse enregistrée lors de l'appariement */
curConnUrl = remBtDevice.getConnectionUrl();
/** Crée une nouvelle connexion */
connection = new BTconn(curConnUrl);
/** Démarre le processus de connexion*/
if (connection.connect(curConnUrl)) {
    if (showView) {
        /* Si la connexion est réussie, retourne au menu de démarrage*/
        showMainMenu();
    }
} else {
    showAlert("Connection Error",
    "impossible to connect to your bluetooth remote device",
    true);
}
```

Code 6 : Connexion Bluetooth client

La méthode boolean connect(String url) permet d'établir la connexion et d'ouvrir les flux d'entrée-sortie de la manière suivante :

```
/*Définition des variables :*/
protected DataInputStream   in          = null;
protected DataOutputStream  out         = null;
public  StreamConnection        sppConn     = null;
if (open()){
       in  = sppConn.openDataInputStream();
       out = sppConn.openDataOutputStream();
}
```

Code 7 : Flux d'entrée / sortie

La méthode `boolean open(…)` essaie d'ouvrir la connexion avec le module Casira de la manière suivante :

```
sppConn = (StreamConnection) Connector.open(connUrl, Connector.READ_WRITE)
```

Lorsque la connexion est établie, les flux sont créés. La connexion est ouverte. Il est possible dès cet instant d'envoyer des messages au module Casira.

I.3. Serveur Bluetooth

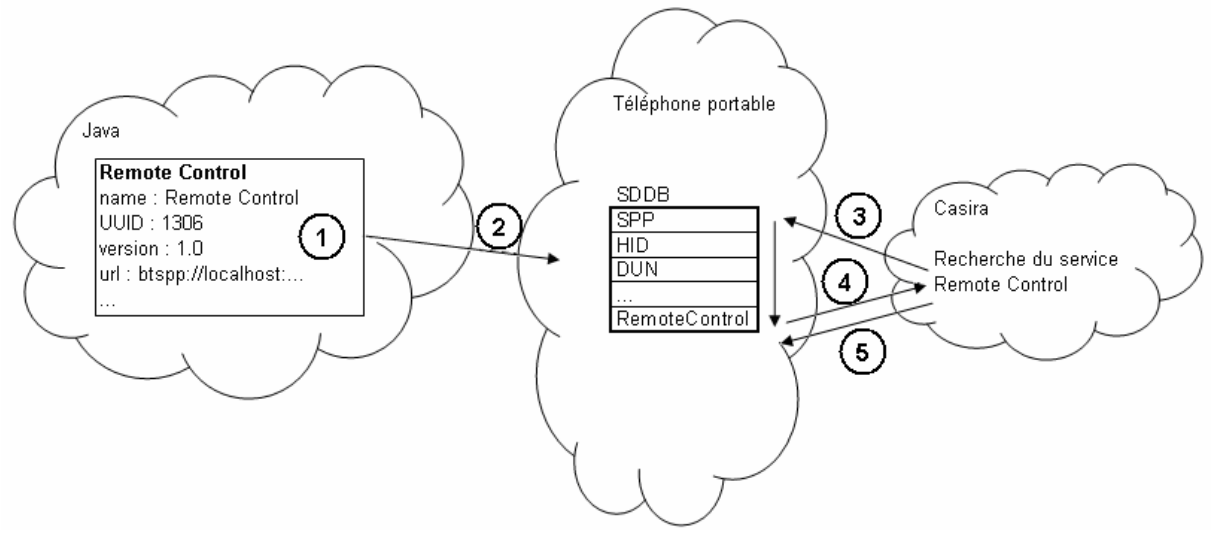


Figure 49 : Serveur Bluetooth

La création d'un serveur en Java permet de mettre à disposition un service propriétaire qui peut être atteint par l'objet intelligent. La procédure depuis la création du serveur à la connexion peut être découpée en cinq grandes étapes :

1. Création du « service record »
2. Enregistrement du service record et démarrage du serveur
3. Recherche du service propriétaire par le Casira
4. Le téléphone indique que le service existe
5. Demande de connexion par le Casira

« Service record » est le terme qui défini un profil Bluetooth. Il contient diverses informations tel que :

- le nom
- l'identifiant
- la version
- l'adresse pour l'atteindre
- si l'appariement est nécessaire ou non
- …

La SDDB est une base de donnée qui contient la liste de tous les profils mis à disposition par un appareil Bluetooth.

L'UUID est un identifiant unique pour chaque profil déclaré. La recherche qui détermine si un profil est accessible ou non se fait grâce à ceci. La norme Bluetooth décrit une grande quantité d'UUID, mais permet également d'en créer de nouveaux.

Création du « service record » et enregistrement

Lorsqu'un avertisseur de connexion est déclaré dans l'application java, le « service record » est créé. L'avertisseur de connexion est du type `StreamConnectionNotifier`

Chaque service Bluetooth et ses attributs sont uniques. La première étape de la création du service est de lui assigner un UUID. Je vais utiliser l'UUID 0x1306 qui est laissé libre par la norme Bluetooth.

```
// Bluetooth service Name
private String appName = "Remote Control";
// Bluetooth unique service UUID
private UUID uuid = new UUID(0x1306);
```

Code 8 : UUID

La prochaine étape définit et crée l'avertisseur de connexion. Ceci produit la création du service record :

```
String connString ="btspp://localhost:" + uuid.toString() +
                   ";name=" + appName;
private StreamConnectionNotifier _service;
_service = (StreamConnectionNotifier)Connector.open(connString);
```

Code 9 : Url serveur et Service Record

Une fois que l'avertisseur de connexion et le service record sont créés, le serveur est prêt à enregistrer le service et accepter une connexion. En appelant la méthode `AcceptAndOpen()` le service record est ajouté dans la SDDB (Service Discovery Data Base). Cette étape rend le service visible par les appareils clients. La méthode `AcceptAndOpen()`est bloquante tant qu'une connexion n'est pas établie.

```
private StreamConnection _rfConn;
// Insert service record into SDDB and wait for incoming connection
_rfConn = _service.acceptAndOpen();
```

Code 10 : acceptAndOpen()

Attente et traitement de la connexion entrante

Il ne reste plus qu'à attendre que l'objet intelligent essaie de se connecter. Lorsque la connexion est établie, l'ouvrir les flux d'entrée/sortie est exécutée.
Tout est en place pour commencer l'échange de messages entre les deux appareils.

```
DataInputStream _in = _rfConn.openDataInputStream()
DataOutputStream _out = _rfConn.openDataOutputStream()
```

Code 11 : Flux d'entrée / sortie

I.4. Envoi de message

L'envoi de messages est réalisé de la même manière entre l'application Java capable d'effectuer les mesures et celle qui dialogue avec l'objet intelligent. L'envoi d'un message se fait sous cette forme :

```
    public boolean send( String toSend ) {
        // Indicate if the sending is successful
        boolean isSent = false;

        //Try to send the message
        try {
                synchronized (out) {
                out.writeChars( toSend );
                out.flush();
                isSent = true;
            }
        }
        //an error appeared
        catch (Exception e) { }

       return isSent;
    }
```

Code 12 : Envoi d'un message

J. Code C ANSI

Il est possible que des parties de codes ci-dessous diffèrent du code complet. Ceci vient du fait que les objets intelligents déclarés sont disponibles sur la maquette en version allégée. Par exemple le garage ne dispose pas de fin de course. Il n'est pas possible de déterminer l'état d'un objet sans avoir de retour de sa part. De ce fait, l'état du garage est simulé et envoyé au téléphone portable.

J.1. Menu de l'objet intelligent

Seul le contenu des fichiers menu.c et menu.h suffit à être adapté d'un objet intelligent à l'autre. Ces deux fichiers contiennent toutes les caractéristiques d'un objet, à savoir :

- Le type d'objet
- Le nom
- Les entrées et sorties câblées
- Les types de sorties
- Les actions à effectuer
- La liste des propriétés
- La liste des comportements

Le type d'objet

Certains objets (P.ex. le garage) sont définis de manière très générique à cause de la pauvreté des possibilités offertes par la norme pour définir des appareils « exotiques » dans le monde du Bluetooth actuel. La définition d'une télévision est nettement plus accessible :

```
/* Defined a television according to the Bluetooth specification */
#define CLASS_OF_DEVICE         0x04043C
```

Le nom

```
#define LOCAL_NAME              "Garage"
#define LOCAL_NAME_LENGTH       6
```

Les entrées et sorties câblées

Il y a au total 8entrées/sorties à configurer.

```
#define PIO2_DIR    OUTPUT
#define PIO3_DIR    INPUT
```

Un maximum de 8 comportements et 4 propriétés sont disponibles avec la carte électronique qui est reliée à l'objet (Annexe B.6). Les comportements de l'objet sont envoyés en série grâce à une seule sortie. Pour bénéficier de plus de possibilités, il est tout à fait possible d'insérer le module amovible Bluetooth sur une carte électronique disposant de plus d'entrées/sorties.

Les types de sorties

Il existe actuellement deux types de commandes générées par l'objet intelligent :

- ON_OFF : simule un interrupteur permanent
- PULSE : simule un bouton poussoir

La durée d'une impulsion est paramétrée en millisecondes à l'aide de ce champ.

```
#define PULSE_DELAY             250
```

Les actions à effectuer

L'objet intelligent est capable de traiter des commandes en fonction de deux types d'entrées :

- Bluetooth
- les entrées câblées

Les commandes reçues par la connexion Bluetooth assignent de nouvelles valeurs aux sorties reliées à l'objet. Actuellement, le changement d'une entrée câblée produit l'envoi d'un message de mise à jour au téléphone, cependant, il est tout à fait possible de modifier les sorties également.

En function de Bluetooth

La méthode `rclSetOutput` est appelée chaque fois qu'une commande est reçue du téléphne portable. Le nombre de `case` à implémenter dépend du nombre de comportements mis à dispositions par l'objet intelligent.

La méthode `sendSer` permet d'assigner de nouvelles valeurs aux différentes sorties. Elle prend comme paramètre la valeur de sortie et le masque qui sélectionne la ou les sorties.

```
void rclSetOutput(uint8 output, uint8 state, uint8 type)
{
    switch(output)
    {
    /* open the door if it's closed*/
    case 0:
        if(door == FALSE)
        {
            sendSer(output+0x01,0x03);
        }
        break;

    /* close the door if it's opened*/
    case 1:
        if(door == TRUE)
        {
            sendSer(output+0x01,0x03);
        }
        break;

    /* Light : toggle */
    case 2:
        …
        break;
    };

    /* Send witch output must be cleared after the pulse delay */
    if(type == PULSE)
    {
        outputToUpdate = malloc(sizeof(uint8));
        *outputToUpdate = output;

        MessageSendLater(getAppTask(), RCL_RESET_OUTPUT, outputToUpdate,
                         PULSE_DELAY);
    }
}
```

Code 13 : Comportements d'un objet en fonction de Bluetooth

En fonction des entrées câblées

La méthode `sendUpdateInput` permet d'envoyer les propriétés de l'objet intelligent lorsque l'un d'entre eux est modifié. Dans cet exemple, l'état de l'objet est modifié en fonction de l'entrée `PIO4`.

```
void sendUpdateInput(void)
{
    uint8 i=0;
    uint8 size_of_menu = sizeof(inputMenu)/sizeof(lineOfInputMenu);
    /* Assign a new value of properties according to an entry */
    if((PioGet() & PIO4) != 0)
    {
        inputMenu[1].current = TRUE;
    }
    else
    {
        inputMenu[1].current = FALSE;
    }

   /* Send the content of the properties to the phone*/
    …
}
```

Code 14 : Comportements d'un objet en fonction des entrées câblées

Les propriétés

Chaque propriété est définie d'après une structure :

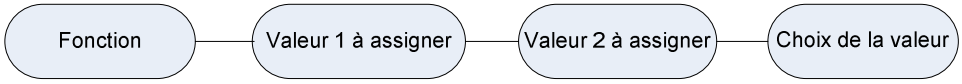


Figure 50 : Structure d'une propriété

Le choix de la valeur est utilisé pour déterminer quel état doit être envoyé à l'interface utilisateur. Un tableau de ces structures permet de crée le menu complet :

```
lineOfInputMenu inputMenu[] = {

    {"Door is ",        "open", "closed",FALSE},
    {"Garage is ",      "empty","full",FALSE},
    {"Light is ",       "on",   "off",  FALSE}
};
```

Les comportements

Chaque comportement est défini d'après une structure :

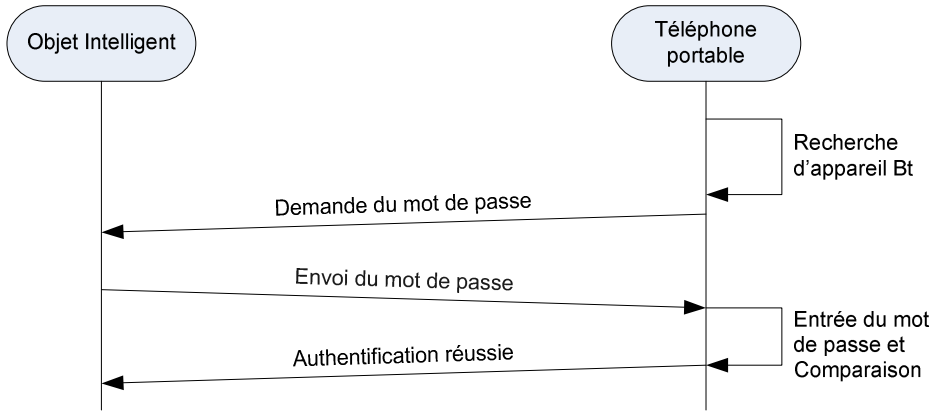
Figure 51 : Structure d'un comportement

Un tableau de ces structures permet de crée le menu complet :

```
static lineOfOutputMenu outputMenu[] = {

    {"Open Door",       PULSE,  ON},
    {"Close Door",      PULSE,  ON},
    {"Toggle the light",ON_OFF, ON}
};
```

J.2. Gestion de l'appariement



À la réception de demande du mot de passe, le message `CL_SM_PIN_CODE_IND` est reçu par l'objet intelligent. Il s'en suit l'envoi du message contenant le mot de passe grâce à la fonction `ConnectionSmPinCodeResponse(const bdaddr *bd_addr, uint16 size_pin_code, const uint8 *pin_code)`. A ce moment là, le téléphone demande à l'utilisateur d'entrer un mot de passe. Si les deux correspondent, les appareils sont dits appariés, une connexion est possible. L'adresse Bluetooth et le nom de l'appareil appariés sont sauvegardés en mémoire.

L'emplacement mémoire a cette structure :

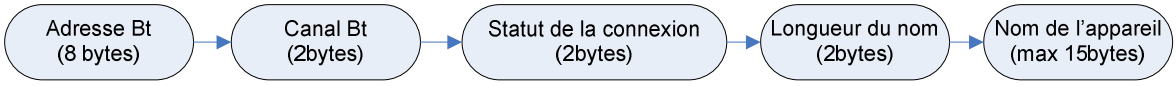


Figure 52 : Structure d'un emplacement mémoire

Le statut de la connexion n'indique pas si la connexion est active ou inactive, mais montre si la connexion avec l'appareil a déjà été effectuée. Ce champ est uniquement une indication mais n'est pas utilisé dans l'application.

La limitation des appareils sauvegardée est fixée à 10 actuellement. Cette limite a été décidée en estimant qu'actuellement un foyer ne compte pas plus de dix téléphones portables utilisés régulièrement.

J.3. Recherche d'un service spécifique

La recherche d'un service spécifique se fait en deux étapes, il faut définir :

- le service à rechercher
- où la recherche s'effectue

La recherche d'un service se fait grâce à l'identifiant unique du service (UUID).

La définition du service à rechercher s'effectue de cette manière :

```
static uint8 sppServiceRequest [] =
{
    0x35,                       /* type = DataElSeq */
    0x03,                       /* size ...3 bytes in DataElSeq */
    0x19, 0x13, 0x06            /* 2 byte UUID 1306 = Remote Control*/
}
```

Code 15 : Déclaration du service à rechercher

Tous les services disponibles forment une petite base de donnée (SDDB). Il est possible de la consulter grâce à une méthode mise à disposition par les libraires du Casira qui permet d'utiliser le profil SDP (voir annexe A.3.2). Ce profil permet de récupérer des informations tel que l'UUID, la version du service, si le service est disponible...

La norme Bluetooth défini les attributs d'un profil, on y trouve notamment :

Attribute Name	Attribute ID	Attribute Value Type
ServiceClassIDList	0x0001	DATSEQ of UUIDs
ServiceRecordState	0x0002	32-bit unsigned integer
ProtocolDescriptorList	**0x0004**	**DATSEQ of DATSEQ of UUID and optional parameters**
LanguageBasedAttributeIDList	0x0006	DATSEQ of DATSEQ triples
ServiceAvailability	0x0008	8-bit unsigned integer
VersionNumberList	0x0200	DATSEQ of 16-bit unsigned integers

L'attribut accessible grâce à l'ID 0x0004 contient une liste des services mis à disposition par un appareil Bluetooth. Il faut donc effectuer la recherche du service à cet emplacement.

```
/*
 * Remote Control Profile Attribute request array
 */
static const uint8 sppAttributeRequest [] =
{
    0x35,                   /* type = DataElSeq */
    0x03,                   /* size ...3 bytes in DataElSeq */
    0x09, 0x00, 0x04        /* 2 byte UINT attrID ProtocolDescriptorList*/
};
```

Code 16 : Emplacement de l'UUID du service

La méthode qui permet d'effectuer la recherche a besoin des paramètres suivants :

- l'adresse Bluetooth du serveur
- le service à rechercher
- le lieu où s'effectue la recherche
- divers autres paramètres

```
ConnectionSdpServiceSearchAttributeRequest (
Task  theAppTask, const bdaddr *  addr,uint16  max_attributes,
uint16  size_search_pattern, const uint8 *  search_pattern,
uint16  size_attribute_list, const uint8 *  attribute_list ).
```

Cette méthode produit un message du type `CL_SDP_SERVICE_SEARCH_ATTRIBUTE_CFM` qui contient le statut de la requête, divers champs ainsi que le canal de l'appareil distant sur lequel le profile Bluetooth est mis à disposition. En récupérant le statut, on détermine si la requête s'est déroulée correctement. Si c'est le cas, il est possible d'établir la connexion entre deux appareils Bluetooth.